



UNIVERSIDAD DE GRANADA

Facultad de Ciencias y ETSIIT

GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS

TRABAJO DE FIN DE GRADO

Análisis de la eficiencia de redes neuronales a través de la anchura de sus capas

Presentado por:
Luis Crespo Ortí

Curso académico 2023-2024

Análisis de la eficiencia de redes neuronales a través de la anchura de sus capas

Luis Crespo Ortí

Luis Crespo Ortí *Análisis de la eficiencia de redes neuronales a través de la anchura de sus capas* .
Trabajo de fin de Grado. Curso académico 2023-2024.

**Responsables de
tutorización**

Francisco Javier Meri de la Maza
Departamento de Análisis Matemático

Grado en Ingeniería
Informática y Matemáticas

Siham Tabik Ouled Hrour
*Departamento de Ciencias de la Computación
e Inteligencia Artificial*

Facultad de Ciencias y
ETSIIT
Universidad de Granada

DECLARACIÓN DE ORIGINALIDAD

D./Dña. Luis Crespo Ortí

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

En Granada a 15 de julio de 2024

Fdo: Luis Crespo Ortí

*A mis padres, mis compañeros y mis profesores,
por acompañarme en esta maravillosa travesía.*

Análisis de la eficiencia de redes neuronales a través de la anchura de sus capas

Luis Crespo Ortí

Palabras clave:

Redes Neuronales, Redes Neuronales Totalmente Conectadas (FNN), Teorema de Aproximación Universal, Redes ReLU, Anchura, Profundidad, Potencia Expresiva, Funciones Integrables, Funciones Lipschitzianas

Resumen

En el siguiente trabajo, se realiza un análisis global de varios aspectos relacionados con la arquitectura de las redes neuronales.

Nos centraremos en las Redes Neuronales Totalmente Conectadas, que constituyen la raíz de todo el avance posterior en el campo. Se analizarán varios Teoremas de Aproximación Universal desde diferentes perspectivas.

El primero de ellos es el *Teorema de Cybenko*, un resultado clásico que nos da una intuición sobre la potencia expresiva de las redes neuronales que cuentan únicamente con una capa oculta en su arquitectura.

Más tarde, analizaremos el *Teorema de Aproximación Universal para redes ReLU limitadas por anchura*, que si bien no acota el número de capas, esto es, su profundidad; sí que establece un límite en el número de neuronas de las mismas, es decir, en su anchura, en función del tamaño de la entrada. Además, utiliza la función de activación ReLU, de amplia popularidad en la actualidad, y la noción de aproximación en el espacio \mathcal{L}^1 de las funciones integrables, la cual desarrollamos constructivamente partiendo de los fundamentos.

Poniendo en común todas las reflexiones anteriores, junto con algunos resultados más, formulamos algunos argumentos preliminares sobre el posible rol de la anchura y la profundidad en las redes neuronales, y la eficiencia de las distintas disposiciones de las neuronas. También identificamos los casos en los que podría producirse una pérdida sustancial de expresividad.

Para finalizar, conducimos algunos experimentos en los que se hace uso de las arquitecturas construidas en las demostraciones de los teoremas. Establecemos condiciones concretas en las que es plausible implementarlas, y hacemos un análisis de sus ventajas y desventajas. Estudiamos su funcionamiento y su capacidad de generalización en entornos controlados definidos por funciones lipschitzianas, utilizando algoritmos iterativos de optimización de parámetros, y distintas inicializaciones de dichos parámetros. Esta línea de investigación nos llevará a extraer una serie de conclusiones muy interesantes que fortalecerán nuestro entendimiento de las redes neuronales y reafirmarán empíricamente algunas de las ideas expuestas anteriormente.

El trabajo se plantea en la búsqueda de puntos en común entre la Informática y las Matemáticas, tratando de ofrecer un marco teórico sólido de análisis que pueda dar luz a futuras investigaciones y mejorar las arquitecturas existentes.

Analysis of the efficiency of neural networks through the width of their layers

Luis Crespo Orti

Keywords:

Neural Networks, Fully Connected Neural Networks (FNN), Universal Approximation Theorem, ReLU Networks, Width, Depth, Expressive Power, Lebesgue-integrable Functions, Lipschitz Functions

Abstract

In this work, a comprehensive analysis of various aspects related to the architecture of neural networks is carried out.

We will focus on Fully Connected Neural Networks, which constitute the root of all subsequent advances in the field. Various Universal Approximation Theorems will be analyzed from different perspectives.

The first of these is the *Cybenko's Theorem*, a classic result that gives us insight into the expressive power of neural networks that have only one hidden layer in their architecture. Later, we will analyze the *Universal Approximation Theorem for Width-Bounded ReLU Networks*, which, although it does not limit the number of layers (i.e., their depth), it does set a limit on the number of neurons in them, that is, their width, based on the size of the input.

Additionally, it uses the ReLU activation function, which is widely popular today, and the notion of approximation in the \mathcal{L}^1 space of Lebesgue-integrable functions, which we develop constructively from the fundamentals.

Bringing together all the previous reflections, along with some additional results, we elaborate some preliminary arguments about the possible role of width and depth in neural networks, and the efficiency of the various neuron arrangements. We also identify cases where a substantial loss of expressiveness could occur.

To conclude, we conducted some experiments using the architectures built in the theorem demonstrations. We established specific conditions under which it is feasible to implement them and analyzed their advantages and disadvantages. We studied their performance and generalization capacity in controlled environments defined by Lipschitz functions, using iterative parameter optimization algorithms and different initializations of these parameters.

This line of research will lead us to draw a series of very interesting conclusions that will strengthen our understanding of neural networks and empirically reaffirm some of the ideas previously discussed.

The work is aimed at finding common ground between Computer Science and Mathematics, trying to offer a solid theoretical analysis framework that can shed light on future research and improve existing architectures.

Índice general

Introducción	xI
Glosario	xV
1 Fundamentos Matemáticos	1
1.1 Intervalos de \mathbb{R}^n	1
1.1.1 Definiciones básicas	1
1.1.2 Cubos diádicos	1
1.2 Medida exterior de Lebesgue	3
1.2.1 La medida elemental de los intervalos acotados	3
1.2.2 La medida exterior	4
1.2.3 Propiedades básicas	6
1.2.4 Medida exterior nula e infinita	7
1.3 Conjuntos medibles de \mathbb{R}^n	8
1.3.1 σ -álgebra de los conjuntos medibles	8
1.3.2 Medibilidad de los intervalos y los abiertos de \mathbb{R}^n	10
1.3.3 Caracterización de la medibilidad	10
1.4 Medida de Lebesgue	11
1.4.1 Propiedades fundamentales	11
1.4.2 Propiedades geométricas	12
1.5 Funciones reales medibles	15
1.5.1 El espacio $\mathcal{L}(\Omega)$	15
1.5.2 El conjunto $\mathcal{L}^+(\Omega)$	16
1.6 Integral de Lebesgue	17
1.6.1 Gráficas y subgráficas	17
1.6.2 La integral y sus propiedades	19
1.6.3 Integrabilidad: el espacio $\mathcal{L}^1(\Omega)$	23
1.7 Estimación mediante funciones simples	23
1.7.1 Distancia en $\mathcal{L}^1(\Omega)$	23
1.7.2 Funciones integrables	24
1.7.3 Funciones continuas	26
1.7.4 Funciones lipschitzianas	27
2 Fundamentos de Redes Neuronales	29
2.1 Componentes de una red neuronal	29
2.1.1 Neuronas	29
2.1.2 Pesos y biases	29
2.1.3 Capas	29
2.1.4 Funciones de activación	30
2.2 Redes Neuronales Totalmente Conectadas	31
2.2.1 Representación matemática	31
2.2.2 Representación gráfica	32

Índice general

2.3	Aprendizaje automático en redes neuronales	33
2.3.1	Funciones de pérdida	34
2.3.2	Metodología de entrenamiento	35
2.3.3	Generalización en redes neuronales	39
3	Capacidad de Cálculo de las Redes Neuronales	45
3.1	Redes acotadas en profundidad	45
3.2	Redes acotadas en anchura	49
3.2.1	Aproximación de la función característica de un cubo	50
3.2.2	Aproximación de una combinación lineal de funciones características de cubos	55
3.2.3	Teorema de Aproximación Universal para redes ReLU limitadas por anchura	57
3.2.4	Pérdida de expresividad de las redes ReLU limitadas por anchura	58
4	Construcción Explícita de Redes Neuronales	61
4.1	Funciones de interés	61
4.2	Determinación de una partición	62
4.2.1	Cálculo del error	63
4.2.2	Algoritmo adaptativo para la minimización del error	64
4.3	Implementación de las redes	68
4.4	Estudio de la parametrización inicial	71
4.4.1	Procedimiento experimental	71
4.4.2	Arquitecturas empleadas	72
4.4.3	Resultados obtenidos	73
4.4.4	Interpretación de las métricas	73
4.4.5	Interpretación de las curvas de aprendizaje	76
4.4.6	Conclusiones generales	77
5	Comentarios Finales	81
5.1	Metodología, organización y planificación	81
5.2	Objetivos cumplidos	84
5.3	Trabajo futuro	84
	Bibliografía	87

Introducción

En la era digital actual, los **datos** han adquirido una importancia sin precedentes. Cada acción que realizamos, desde la navegación en internet hasta las transacciones financieras, genera una enorme cantidad de datos. Estos datos suelen presentar patrones sofisticados, cuyo correcto estudio e interpretación tiene el potencial de revolucionar industrias enteras, optimizar procesos y mejorar la toma de decisiones en innumerables áreas. La capacidad para extraer información valiosa de los datos se ha convertido en un recurso fundamental para la competitividad y la innovación en el mundo moderno.

En este contexto, la inteligencia artificial, el aprendizaje automático y, en concreto, las **redes neuronales**, emergen como herramientas poderosas para abordar muchos de los desafíos relacionados con el análisis de grandes volúmenes de datos. Estas estructuras, asistidas mediante refinados algoritmos de optimización, han demostrado ser sorprendentemente eficaces en el reconocimiento de características complejas. Esto permite realizar predicciones precisas y ofrecer soluciones a problemas que antes parecían insuperables. Los resultados obtenidos a través de las redes neuronales son prometedores y abarcan desde el reconocimiento de voz y la traducción automática, hasta la detección de enfermedades y la conducción autónoma.

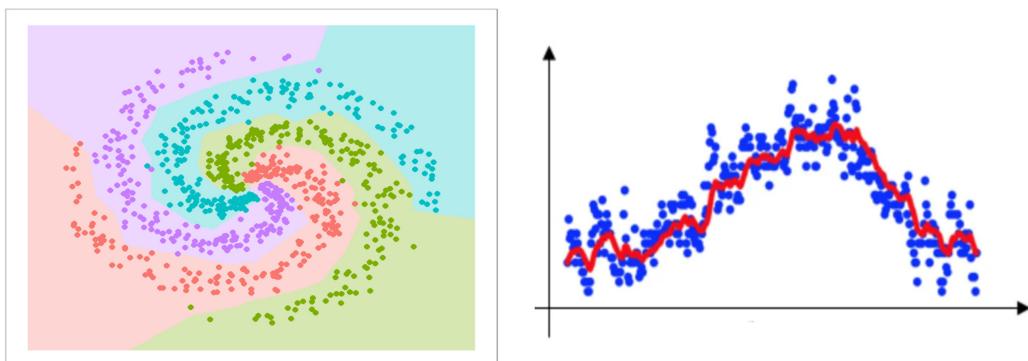


Figura 1: Ejemplo de distribuciones de datos, donde cada uno es un punto en el espacio. En la distribución de la izquierda, una red neuronal tendría que modelizar las fronteras de decisión (*decision boundaries*) [6] que determinan a qué clase pertenecerá cualquier dato considerado (clasificación). Para la de la derecha, la tarea de una red sería ajustarse a la forma de los datos de forma que pueda realizar predicciones (regresión). Imágenes extraídas de [1] y [8] y posteriormente modificadas.

A pesar de estos sorprendentes avances, aún hay muchas incógnitas y áreas por explorar dentro del campo de las redes neuronales. Aunque la producción de artículos científicos sobre el tema está en auge y la tendencia global es la búsqueda del *estado del arte*, también resulta fundamental plantear análisis más elementales. Los modelos neuronales suelen recibir el nombre de **cajas negras** precisamente por su poca transparencia y la dificultad para interpretar cómo y por qué toman ciertas decisiones. Para desentrañar los misterios de su funcionamiento, resulta necesario regresar a las configuraciones más básicas y estudiar sus

fundamentos teóricos y prácticos con mayor detenimiento. Al contrario de lo que pueda parecer, este estudio puede resultar muy complejo, pero a través de su exploración minuciosa podremos aprovechar plenamente el potencial de las redes neuronales para desplegarlo en diversas aplicaciones.

En este sentido, uno de los análisis más relevantes es el de la capacidad teórica de **aproximación** de las redes neuronales. Esta línea de investigación supone un punto de confluencia significativo entre las Matemáticas y la Informática, estudiando la potencia de las redes neuronales para estimar funciones con ciertas propiedades, que en la práctica se pueden interpretar como la distribución de los datos que tratamos de modelar. En el [Capítulo 3](#), realizamos un extenso análisis de algunos de estos resultados, examinando sus requisitos concretos y desarrollando intuiciones al respecto.

El conocido como *Teorema de Cybenko* [13] es uno de los resultados precursores en este ámbito, y establece condiciones muy generales para la aproximación por parte de redes neuronales de escasa complejidad estructural. No obstante, el exponencial avance paralelo en el terreno empírico deja algunas de las suposiciones y técnicas consideradas como obsoletas, y urge la necesidad de proponer enfoques más adaptados a los nuevos tiempos, y que exploren métodos más modernos. En esta línea, surgen resultados como el *Teorema de Aproximación Universal para redes ReLU limitadas por anchura*, que fue planteado y demostrado por unos investigadores de la Universidad de Pekín en 2017, en el artículo [37]. Este teorema nos da una perspectiva de cómo redes neuronales que usan métodos más adecuados al proceder experimental actual también poseen esa capacidad expresiva. Además, el punto más interesante es que gozan de esta competencia pudiendo acotar, hasta cierto punto, uno de sus parámetros fundamentales: la anchura. Este detalle contrasta con el Teorema de Cybenko que acota, precisamente, otro de los elementos relevantes de la arquitectura de las redes: la profundidad.

Por otro lado, también se modifica la noción de aproximación de este segundo teorema con respecto del primero. En este caso, no aproximamos una función de referencia uniformemente en cada punto, sino que se demuestra la posibilidad de minimización de la **distancia en \mathcal{L}^1** de cualquier función integrable con respecto de una red neuronal diseñada para tal propósito. Para entender con detalle el significado de este concepto, en el [Capítulo 1](#), se desarrolla una introducción matemática del mismo de manera constructiva y natural. Específicamente, se introduce la medida de Lebesgue, y posteriormente, se define la integral de Lebesgue de una función medible positiva como la medida de la subgráfica de dicha función. De esta manera, se extienden satisfactoriamente las nociones intuitivas en una, dos y tres dimensiones a un mayor número de ellas. Se dedica un gran esfuerzo a asegurar la precisión y rigurosidad de los pormenores de esta exposición. Así mismo, en el [Capítulo 2](#) explicamos brevemente el tratamiento matemático de las redes neuronales, para poder abordar las demostraciones de los teoremas convenientemente.

Una vez hemos constatado la potencia expresiva de las redes neuronales con **profundidad** o **anchura** acotadas, cabe preguntarse en qué condiciones concretas esta expresividad se puede ver disminuida. Esto nos conduce a reflexiones sobre la relación entre ambos factores y qué lugar ocupan en una arquitectura neuronal, evaluando cuál de ellos podría ser más eficiente en términos relativos. En la [Subsección 3.2.4](#), se lleva a cabo una discusión al respecto utilizando numerosas referencias que apuntan en esta dirección.

Por último, trasladamos todos estos argumentos a la **práctica**, observando qué utilidad podrían tener, detectando restricciones no consideradas anteriormente y validando algunas de nuestras hipótesis. Específicamente, en el [Capítulo 4](#), se implementan las redes descritas

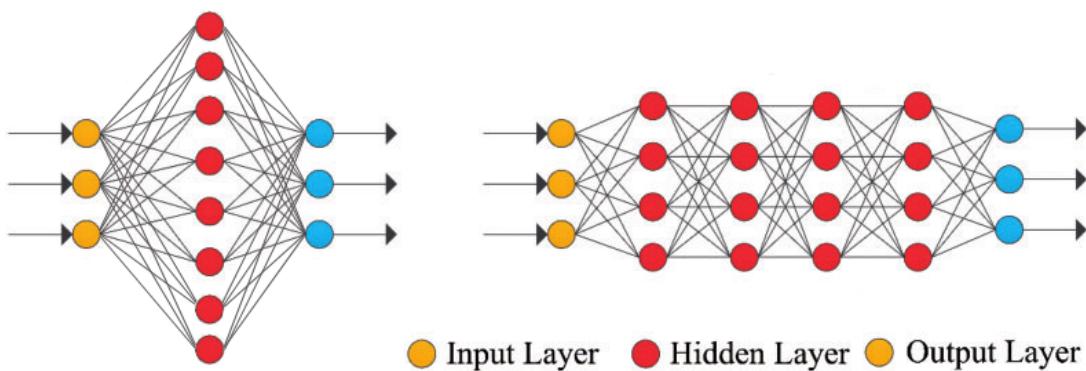


Figura 2: Diagrama de una red neuronal más ancha y menos profunda (a la izquierda) y una red menos ancha y más profunda (a la derecha). Imagen extraída de [4] y posteriormente modificada.

en la [Sección 3.1](#) y la [Sección 3.2](#), y se analizan en detalle las características de su estructura, así como su rendimiento cuando se encadenan con procedimientos de optimización estándar que se describen en la [Sección 2.3](#). El estudio se realiza desde el punto de vista de las **funciones lipschitzianas**, que establecen un marco coherente donde los resultados teóricos se pueden aplicar con cierto sentido. Bajo estas premisas, se examinan las capacidades de ajuste y generalización de cada modelo, llegando a conclusiones muy reveladoras.

Por tanto, el objetivo del trabajo será profundizar en la comprensión de las redes neuronales, tanto desde una perspectiva teórica como práctica. Este enfoque integral nos permitirá maximizar su potencial, identificar y superar sus limitaciones, y explorar nuevas posibilidades de aplicación. Con esta base sólida en mente, estaremos mejor preparados para adaptarnos al incesante avance de estas tecnologías, extender nuestras intuiciones a ellas y enfrentar con garantías los retos del futuro.

Glosario

\mathbb{R}_0^+ Conjunto de los reales positivos y el cero. $\mathbb{R}_0^+ = \{x \in \mathbb{R} : x \geq 0\}$

\mathbb{N}_0 Conjunto de los números naturales junto con el cero. $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

Intervalo de \mathbb{R} Conjunto $I \subseteq \mathbb{R}$ que verifica que si $r, t \in I$ con $r \leq t$, entonces $s \in \mathbb{R}$ con $r \leq s \leq t$ verifica que $s \in I$. Esta definición incluye al conjunto vacío \emptyset y a todos los conjuntos $\{x\}$ con $x \in \mathbb{R}$, a los que llamamos **intervalos degenerados**.

Γ_n Conjunto de los primeros n números naturales. Esto es, $\Gamma_n = \{k \in \mathbb{N} \mid k \leq n\} = \{1, 2, \dots, n\}$

$\mathcal{P}(A)$ Partes de un conjunto A . Es el conjunto de subconjuntos de A , incluyendo \emptyset y A .

\sqcup Simboliza una unión disjunta. No se utiliza para uniones casi disjuntas.

\triangle Unión de las diferencias entre dos conjuntos. $A \triangle B = (A \setminus B) \cup (B \setminus A)$.

$\text{int}(A)$ Interior del conjunto A .

$\text{cl}(A)$ Clausura del conjunto A .

G_δ Tipo de conjunto de \mathbb{R}^n que se construye como una intersección numerable de abiertos de \mathbb{R}^n .

1 Fundamentos Matemáticos

A continuación, presentamos toda la teoría matemática necesaria para abordar los enunciados y demostraciones del Capítulo 3, así como la argumentación de la implementación planteada en el Capítulo 4.

Para la presentación, se han empleado principalmente los siguientes libros: [5], [47], [58], [56], [48] y [19]. Estos recursos se han adaptado y combinado adecuadamente para ofrecer una exposición clara y compacta. Algunas de las demostraciones se han desarrollado de manera independiente, en ocasiones con inspiración en algunos de los resultados de dichos libros.

En lo sucesivo, trabajaremos en el contexto de \mathbb{R}^n , para un $n \in \mathbb{N}$ fijo.

1.1. Intervalos de \mathbb{R}^n

1.1.1. Definiciones básicas

Definición 1.1.1. Un **intervalo** $J \subset \mathbb{R}^n$ es la extensión natural del concepto de intervalo conocido para \mathbb{R} . Se puede definir como un producto cartesiano $J = J_1 \times J_2 \times \cdots \times J_n$, donde J_i es un intervalo de \mathbb{R} para cada $i \in \{1, \dots, n\} = \Gamma_n$.

Definición 1.1.2. Decimos que el intervalo $J \subset \mathbb{R}^n$ es **acotado** si, para cada $i \in \Gamma_n$, la proyección sobre la i -ésima coordenada de J es un intervalo acotado de \mathbb{R} . Notamos por \mathcal{I}_n al conjunto de todos los intervalos acotados de \mathbb{R}^n , o simplemente \mathcal{I} cuando no haya ambigüedad. Asumiremos que $\emptyset \in \mathcal{I}$.

De manera similar, un intervalo de \mathbb{R}^n será **abierto** si proyecta en un intervalo abierto de \mathbb{R} en cada dimensión, y **cerrado** si todas sus proyecciones son intervalos cerrados de \mathbb{R} .

Definición 1.1.3. La **longitud de un intervalo acotado** $J \subset \mathbb{R}$ se define como $\ell(J) = \sup(J) - \inf(J)$. Notamos que dos intervalos de extremos iguales tienen la misma longitud (independientemente de si dichos extremos son cerrados o abiertos) y que los intervalos degenerados tienen longitud cero. Además, tomamos $\ell(\emptyset) = 0$.

Definición 1.1.4. Un **cubo** $C \subset \mathbb{R}^n$ es un intervalo de \mathbb{R}^n cuyas proyecciones unidimensionales son intervalos acotados de igual longitud. Si además todas esas proyecciones son el mismo intervalo $J \subset \mathbb{R}$, lo notaremos por $C = J^n$.

1.1.2. Cubos diádicos

Definición 1.1.5. Sea $k \in \mathbb{N}_0$, definimos

$$C_k = [0, 2^{-k}]^n \quad \text{y} \quad P_k = \{p \in \mathbb{R}^n : 2^k p \in \mathbb{Z}^n\}.$$

Un **cubo diádico** $D \subset \mathbb{R}^n$ de orden k es una traslación de C_k por algún elemento de P_k . Esto es, $D = p + C_k$, con $p \in P_k$.

Proposición 1.1.1. *El conjunto \mathbb{R}^n se puede expresar como una unión disjunta de los cubos diádicos de un orden fijo.*

Demostración. Sean $x \in \mathbb{R}^n$, $k \in \mathbb{N}_0$. Si $x \in D = p + C_k$, con $p \in P_k$, la homotecia de razón 2^k sobre D lleva p en $2^k p \in \mathbb{Z}^n$ y x en $2^k x \in \mathbb{R}^n$. Como $2^k D = 2^k p + [0, 1]^n$, entonces $\pi_i(2^k p) = \lfloor \pi_i(2^k x) \rfloor$ para todo $i \in \Gamma_n$. Por otro lado, la homotecia es una biyección y cada real tiene una única parte entera, por lo que existe un único $p \in P_k$ tal que $x \in p + C_k$. \square

Proposición 1.1.2. *Sean $k, k' \in \mathbb{N}_0$ tal que $k > k'$. Cada cubo diádico de orden k está contenido en un único cubo diádico de orden k' .*

Demostración. Sea $k \in \mathbb{N}$. Por la propiedad anterior, para cada punto $q \in P_k \subset \mathbb{R}^n$, existe un solo $p \in P_{k-1}$ tal que $q \in p + C_{k-1}$. Entonces, tenemos que $s = q - p \in C_{k-1}$, es decir, $0 \leq \pi_i(2^{k-1}s) < 1 \forall i \in \Gamma_n$. Notamos también que $p \in P_k$, ya que $2 \cdot 2^{k-1}p \in \mathbb{Z}^n$, luego $s \in P_k$.

Podemos reescribir la desigualdad anterior como $0 \leq 2^k \pi_i(s) < 2$, y como además $2^k s \in \mathbb{Z}^n$, entonces $2^k s \in \{0, 1\}^n$. Tomando un $x \in C_k$, tenemos que $0 \leq 2^k(\pi_i(s) + \pi_i(x)) < 2 \iff 0 \leq 2^{k-1}\pi_i(s + x) < 1 \forall i \in \Gamma_n$. Esto implica que $s + x \in C_{k-1}$, o lo que es lo mismo, $q + x \in p + C_{k-1}$. Como $x \in C_k$ es arbitrario, concluimos que $q + C_k \subset p + C_{k-1}$. Entonces, cualquier cubo diádico de orden k está contenido en un único cubo diádico de orden $k-1$. Aplicando inducción, es claro que el resultado es válido para cualquier $k' \leq k-1$. \square

Proposición 1.1.3. *Todo abierto de \mathbb{R}^n es unión numerable de cubos diádicos disjuntos.*

Demostración. Sea un abierto $E \subset \mathbb{R}^n$. Definimos los siguientes conjuntos:

$$\tilde{P}_1 = \{p \in P_1 : p + C_1 \subset E\} \quad \text{y} \quad \tilde{D}_1 = \bigsqcup_{p \in \tilde{P}_1} (p + C_1).$$

De forma recursiva, y para $k \in \mathbb{N}$, tomamos

$$\tilde{P}_k = \left\{ p \in P_k : p + C_k \subset E \setminus \left(\bigcup_{j=1}^{k-1} \tilde{D}_j \right) \right\} \quad \text{y} \quad \tilde{D}_k = \bigsqcup_{p \in \tilde{P}_k} (p + C_k).$$

Es claro que \tilde{D}_k es una unión numerable, pues $\tilde{P}_k \subset P_k$ siendo P_k numerable, al poder identificarse con \mathbb{Z}^n . Además, es disjunta, al ser los cubos diádicos de orden k disjuntos dos a dos.

Si $x \in \tilde{D}_k$, entonces $x \in p + C_k \subset E \setminus (\bigcup_{j=1}^{k-1} \tilde{D}_j)$ para cierto $p \in \tilde{P}_k$, por lo que $x \notin \tilde{D}_j$ para ningún $j \in \Gamma_{k-1}$. Por tanto, la familia $\{\tilde{D}_k : k \in \mathbb{N}\}$ es disjunta, y podemos definir

$$\tilde{E} = \bigsqcup_{k=1}^{\infty} \tilde{D}_k$$

como una unión numerable de cubos diádicos disjuntos de todos los órdenes. Vamos a probar que este conjunto \tilde{E} es exactamente el abierto E . La inclusión $\tilde{E} \subseteq E$ es evidente, puesto que todos los cubos diádicos que conforman \tilde{E} están contenidos en E , por la definición de \tilde{P}_k para cada $k \in \mathbb{N}$. Probamos pues la otra inclusión.

Sea $x \in E$, existe $\epsilon > 0$ tal que $B(x, \epsilon) \subset E$, por ser E abierto. Como todas las normas son equivalentes en \mathbb{R}^n , vamos a considerar las bolas utilizando la distancia inducida por la norma del máximo, que denotaremos mediante $\|\cdot\|$.

Tomamos pues $k \in \mathbb{N}$ con $2^{-k} < \frac{\epsilon}{2}$, y encontramos el único cubo diádico de orden k que contiene a x , es decir, $D = p + C_k$ para cierto $p \in P_k$. Para cada $y \in D$, $\|y - p\| < 2^{-k} < \frac{\epsilon}{2}$. Se deduce pues que $\|x - y\| \leq \|x - p\| + \|p - y\| < \epsilon$. Por tanto, $x \in D \subset B(x, \epsilon) \subset E$, es decir, hemos encontrado un cubo diádico D de orden k que contiene a x y está contenido en E . Queda ver si dicho cubo pertenece a \tilde{E} .

Si $p \in \tilde{P}_k$, entonces, $x \in D = p + C_k \subset \tilde{D}_k \subset \tilde{E}$. Por el contrario, si $p \notin \tilde{P}_k$, sabiendo que $p + C_k = D \subset E$, entonces debe existir por definición un cubo diádico D' de orden k' menor que k , de forma que $D' \subset \tilde{D}_{k'}$ y $D \cap D' \neq \emptyset$. Por la [Proposición 1.1.2](#), sabemos que D está contenido en un único cubo diádico de orden k' , el cual debe ser exactamente D' por tener intersección nula con D . Entonces, $x \in D \subset D' \subset \tilde{D}_{k'} \subset \tilde{E}$. En ambos casos, $x \in \tilde{E}$, luego $E \subseteq \tilde{E}$. \square

1.2. Medida exterior de Lebesgue

1.2.1. La medida elemental de los intervalos acotados

Definición 1.2.1. Establecemos la **medida elemental de un intervalo acotado de \mathbb{R}^n** como

$$\begin{aligned} M : \mathcal{I} &\rightarrow \mathbb{R}_0^+ \\ I &\mapsto \prod_{i=1}^n \ell(\pi_i(I)). \end{aligned}$$

Sea $I \in \mathcal{I}$, $M(I)$ es el producto de las longitudes de todos los intervalos de \mathbb{R} que se obtienen al proyectar I en cada una de sus n dimensiones. De esta forma, se generaliza la noción de área de un rectángulo o volumen de un ortoedro para un número mayor de dimensiones.

Definición 1.2.2. Una unión de intervalos acotados de \mathbb{R}^n se dice **casi disjunta** si la intersección entre dos de estos intervalos cualesquiera, que es también un intervalo acotado, tiene medida M nula. Esto sólo ocurre cuando dicha intersección es vacía o un intervalo acotado degenerado.

Proposición 1.2.1. Sea $m \in \mathbb{N}$ y F, F_1, F_2, \dots, F_m intervalos cerrados de \mathbb{R}^n tal que $F \subset \bigcup_{j=1}^m F_j$. Entonces

$$M(F) \leq \sum_{j=1}^m M(F_j).$$

En concreto, si dichos intervalos son casi disjuntos y F es su unión, $M(F) = \sum_{j=1}^m M(F_j)$.

Demostración. Tomamos $F_0 = F$. Sea $i \in \Gamma_n$, consideramos el conjunto X_i que contiene todos los extremos de la i -ésima proyección de todos los intervalos de la familia $\{F_j\}_{j=0}^m$, es decir, $X_i = \{\inf \pi_i(F_j) : j \in \Gamma_m \cup \{0\}\} \cup \{\sup \pi_i(F_j) : j \in \Gamma_m \cup \{0\}\}$. Definimos para cada $i \in \Gamma_n$ y cada $j \in \Gamma_m \cup \{0\}$ el conjunto finito ordenado

$$\Delta_{i,j} = \left(\delta_{i,j,k} \in X_i : \inf \pi_i(F_j) \leq \delta_{i,j,k-1} < \delta_{i,j,k} \leq \sup \pi_i(F_j), k \in \mathbb{N} \right)$$

que contiene los distintos extremos de X_i que se encuentran en la i -ésima proyección del intervalo F_j . Entonces, esta proyección se puede expresar como $\pi_i(F_j) = \bigcup_{k=2}^{|\Delta_{i,j}|} [\delta_{i,j,k-1}, \delta_{i,j,k}]$,

siendo $|\Delta_{i,j}|$ la cardinalidad del conjunto $\Delta_{i,j}$. Además, tenemos que

$$\ell(\pi_i(F_j)) = \sum_{k=2}^{|\Delta_{i,j}|} (\delta_{i,j,k} - \delta_{i,j,k-1}). \quad (1.1)$$

Por tanto, dado un $j \in \Gamma_m \cup \{0\}$, es posible escribir

$$F_j = \bigcup_{k_1=2}^{|\Delta_{1,j}|} \bigcup_{k_2=2}^{|\Delta_{2,j}|} \cdots \bigcup_{k_n=2}^{|\Delta_{n,j}|} Q_{j;k_1,\dots,k_n}$$

donde $Q_{j;k_1,\dots,k_n} = [\delta_{1,j,k_1-1}, \delta_{1,j,k_1}] \times [\delta_{2,j,k_2-1}, \delta_{2,j,k_2}] \times \cdots \times [\delta_{n,j,k_n-1}, \delta_{n,j,k_n}]$, con $2 \leq k_i \leq |\Delta_{i,j}|$, $i \in \Gamma_n$. Atendiendo a la definición del conjunto $\Delta_{i,j}$, vemos que la intersección de dos términos distintos de la unión anterior será un punto en al menos una de las dimensiones, por lo que es un intervalo degenerado. Entonces, la expresión es una unión casi disjunta. Para cada $i \in \Gamma_n$, $\Delta_{i,0} \subset \bigcup_{j=1}^m \Delta_{i,j}$ por ser $F_0 \subset \bigcup_{j=1}^m F_j$, luego se cumple

$$\{Q_{0;k_1,\dots,k_n} : 2 \leq k_i \leq |\Delta_{i,0}|, i \in \Gamma_n\} \subset \{Q_{j;k_1,\dots,k_n} : 2 \leq k_i \leq |\Delta_{i,j}|, i \in \Gamma_n, j \in \Gamma_m\}. \quad (1.2)$$

Usando estas observaciones y (1.1), nos queda que

$$\begin{aligned} M(F) &= M(F_0) = \prod_{i=1}^n \ell(\pi_i(F_0)) = \prod_{i=1}^n \sum_{k=2}^{|\Delta_{i,0}|} (\delta_{i,0,k} - \delta_{i,0,k-1}) \\ &= \sum_{k_1=2}^{|\Delta_{1,0}|} \cdots \sum_{k_n=2}^{|\Delta_{n,0}|} ((\delta_{1,0,k_1} - \delta_{1,0,k_1-1}) \cdots (\delta_{n,0,k_n} - \delta_{n,0,k_n-1})) = \sum_{k_1=2}^{|\Delta_{1,0}|} \cdots \sum_{k_n=2}^{|\Delta_{n,0}|} M(Q_{0;k_1,\dots,k_n}) \\ &\leq \sum_{j=1}^m \sum_{k_1=2}^{|\Delta_{1,j}|} \cdots \sum_{k_n=2}^{|\Delta_{n,j}|} M(Q_{j;k_1,\dots,k_n}) = \sum_{j=1}^m \sum_{k_1=2}^{|\Delta_{1,j}|} \cdots \sum_{k_n=2}^{|\Delta_{n,j}|} ((\delta_{1,j,k_1} - \delta_{1,j,k_1-1}) \cdots (\delta_{n,j,k_n} - \delta_{n,j,k_n-1})) \\ &= \sum_{j=1}^m \prod_{i=1}^n \sum_{k=2}^{|\Delta_{i,j}|} (\delta_{i,j,k} - \delta_{i,j,k-1}) = \sum_{j=1}^m \prod_{i=1}^n \ell(\pi_i(F_j)) = \sum_{j=1}^m M(F_j). \end{aligned} \quad (1.3)$$

Suponiendo que la familia $\{F_j\}_{j=1}^m$ es casi disjunta con $F = \bigcup_{j=1}^m F_j$, entonces podemos escribir la siguiente unión casi disjunta:

$$F = \bigcup_{j=1}^m \bigcup_{k_1=2}^{|\Delta_{1,j}|} \bigcup_{k_2=2}^{|\Delta_{2,j}|} \cdots \bigcup_{k_n=2}^{|\Delta_{n,j}|} Q_{j;k_1,\dots,k_n}.$$

Como se tiene que $\Delta_{i,0} = \bigcup_{j=1}^m \Delta_{i,j} \forall i \in \Gamma_n$, se da la igualdad entre las familias de (1.2), siendo ambas casi disjuntas, lo cual nos permite tomar la igualdad en (1.3). \square

1.2.2. La medida exterior

Podemos tratar de extender la medida elemental de intervalos acotados para poder medir subconjuntos arbitrarios de \mathbb{R}^n .

Definición 1.2.3. La medida exterior de Lebesgue se define como una función $\mu^* : \mathcal{P}(\mathbb{R}^n) \rightarrow$

$\mathbb{R}_0^+ \cup \{\infty\}$ tal que

$$\mu^*(A) = \inf \left\{ \sum_{j=1}^{\infty} M(I_j) : A \subset \bigcup_{j=1}^{\infty} I_j, I_j \in \mathcal{I} \ \forall j \in \mathbb{N} \right\} \quad \forall A \subset \mathbb{R}^n.$$

$\mu^*(A)$ evalúa el ínfimo de las sumas de medidas elementales tomadas sobre intervalos acotados de \mathbb{R}^n que forman un recubrimiento del conjunto $A \subset \mathbb{R}^n$. De alguna manera, considera sucesiones de intervalos que recubren A cada vez con menor exceso de medida.

Para indicar explícitamente el espacio sobre el que medimos, notaremos a la medida exterior por μ_n^* .

Proposición 1.2.2. *La medida exterior de Lebesgue extiende a la medida elemental de los intervalos acotados. Esto es, dado un $I \in \mathcal{I}$, se tiene que $\mu^*(I) = M(I)$.*

Demostración. La desigualdad $\mu^*(I) \leq M(I)$ es evidente, ya que podemos tomar el recubrimiento numerable de I formado por sí mismo y conjuntos vacíos, de forma que la suma de medidas sea exactamente $M(I)$. Para la otra desigualdad, consideramos una familia numerable de intervalos acotados $\{I_j\} \subset \mathcal{I}$ que cubre a I . Para cada $j \in \mathbb{N}$, y dado un $t \in \mathbb{R}^+$, definimos

$$E_j(t) = \prod_{i=1}^n [\inf \pi_i(I_j) - t, \sup \pi_i(I_j) + t]. \quad (1.4)$$

Este conjunto es un intervalo abierto acotado de \mathbb{R}^n . Además, $I_j \subset E_j(t)$. La función $M \circ E_j : \mathbb{R}^+ \rightarrow \mathcal{I}$ es polinómica y estrictamente creciente, y tenemos que $\lim_{t \rightarrow 0} M(E_j(t)) = M(I_j)$. Por tanto, para un $\epsilon > 0$, existe un $\alpha_j > 0$ tal que si $0 < t < \alpha_j$, entonces $M(E_j(t)) < M(I_j) + 2^{-(j+1)}\epsilon$. Elegimos pues $t_j \in]0, \alpha_j[$ para cada $j \in \mathbb{N}$.

Por otro lado, tomando $t \in \mathcal{T} =]0, \min_{i \in \Gamma_n} \{ \frac{\sup \pi_i(I) - \inf \pi_i(I)}{2} \} [$, definimos

$$F(t) = \prod_{i=1}^n [\inf \pi_i(I) + t, \sup \pi_i(I) - t].$$

Este conjunto es un intervalo cerrado de \mathbb{R}^n con $F(t) \subset I$. La función $M \circ F : \mathcal{T} \rightarrow \mathcal{I}$ es polinómica y estrictamente decreciente, con $\lim_{t \rightarrow 0} M(F(t)) = M(I)$. Por tanto, para el $\epsilon > 0$ anterior, existe un $\beta > 0$ tal que si $0 < t < \beta$, entonces $M(I) < M(F(t)) + \epsilon/2$. Tomamos $t_0 \in]0, \beta[$.

Es claro entonces que $F(t_0) \subset I \subset \bigcup_{j=1}^{\infty} I_j \subset \bigcup_{j=1}^{\infty} E_j(t_j)$. Como $F(t_0)$ es compacto, existe $m \in \mathbb{N}$ tal que $F(t_0) \subset \bigcup_{j=1}^m E_j(t_j)$. Denotando por $cl(E_j(t_j))$ al cierre de $E_j(t_j)$ para cada $j \in \Gamma_m$, que es un intervalo cerrado de \mathbb{R}^n con $M(cl(E_j(t_j))) = M(E_j(t_j))$ (ya que los extremos de cada proyección son iguales), vemos que $\{cl(E_j(t_j))\}_{j=1}^m$ recubre el intervalo $F(t_0)$ por intervalos cerrados. Usando la [Proposición 1.2.1](#),

$$\begin{aligned} M(I) &< M(F(t_0)) + \frac{\epsilon}{2} \leq \sum_{j=1}^m M(cl(E_j(t_j))) + \frac{\epsilon}{2} \\ &= \sum_{j=1}^m M(E_j(t_j)) + \frac{\epsilon}{2} < \sum_{j=1}^m (M(I_j) + \frac{\epsilon}{2^{j+1}}) + \frac{\epsilon}{2}. \end{aligned}$$

Tenemos pues que $M(I) < \sum_{j=1}^{\infty} M(I_j) + \epsilon$, al ser $\{I_j\} \subset \mathcal{I}$ y ϵ arbitrarios, se tiene que $M(I) \leq \mu^*(I)$. \square

De ahora en adelante, podemos prescindir de la medida M y utilizar siempre μ^* sobre intervalos acotados de \mathbb{R}^n .

1.2.3. Propiedades básicas

Propiedad 1.2.1 (Monotonía). *Dados $A \subset B \subset \mathbb{R}^n$, tenemos que $\mu^*(A) \leq \mu^*(B)$.*

Demostración. Evidente, por ser cualquier recubrimiento de B un recubrimiento de A . \square

Propiedad 1.2.2 (σ -subaditividad). *Dada una familia numerable de conjuntos $\{A_j\} \subset \mathcal{P}(\mathbb{R}^n)$, se tiene que*

$$\mu^*\left(\bigcup_{j=1}^{\infty} A_j\right) \leq \sum_{j=1}^{\infty} \mu^*(A_j).$$

Demostración. El resultado es trivial si $\mu^*(A_j) = \infty$ para cierto $j \in \mathbb{N}$. En caso contrario, fijando $\epsilon > 0$, podemos construir una familia numerable $\{I_{j,k}\} \subset \mathcal{I}$ para cada $j \in \mathbb{N}$ con

$$A_j \subset \bigcup_{k=1}^{\infty} I_{j,k} \quad \text{y} \quad \sum_{k=1}^{\infty} \mu^*(I_{j,k}) < \mu^*(A_j) + \frac{\epsilon}{2^j},$$

utilizando la definición de ínfimo. La unión de dichas familias para todo $j \in \mathbb{N}$ es obviamente un recubrimiento numerable de $\bigcup_{k=1}^{\infty} A_k$. Por tanto,

$$\mu^*\left(\bigcup_{j=1}^{\infty} A_j\right) \leq \sum_{j=1}^{\infty} \sum_{k=1}^{\infty} \mu^*(I_{j,k}) < \sum_{j=1}^{\infty} \mu^*(A_j) + \epsilon.$$

Esto es equivalente al enunciado por ser $\epsilon > 0$ arbitrario. \square

Observación 1.2.1. De la [Propiedad 1.2.2](#) se deduce la **subaditividad finita**, ya que si $m_0 \in \mathbb{N}$ y $\{A_1, A_2, \dots, A_{m_0}\}$ es una familia finita de conjuntos de \mathbb{R}^n , basta con tomar $A_m = \emptyset \forall m > m_0$, y como lógicamente $\mu^*(\emptyset) = 0$, también se cumple la desigualdad.

Propiedad 1.2.3 (Regularidad exterior). *Sea $A \subset \mathbb{R}^n$,*

$$\mu^*(A) = \inf \{\mu^*(E) : A \subset E, E \text{ abierto de } \mathbb{R}^n\}.$$

Demostración. Sea $\mathcal{F} = \inf \{\mu^*(E) : A \subset E, E \text{ abierto de } \mathbb{R}^n\}$. Tomamos un abierto $E \subset \mathbb{R}^n$ con $A \subset E$. Por la monotonía de μ^* , se tiene que $\mu^*(A) \leq \mu^*(E)$, luego $\mu^*(A) \leq \mathcal{F}$.

La otra desigualdad se cumple trivialmente si $\mu^*(A) = \infty$. Suponiendo $\mu^*(A)$ finito y $\epsilon > 0$, podemos recubrir A mediante una familia de intervalos acotados $\{I_j\} \subset \mathcal{I}$ tal que

$$\sum_{j=1}^{\infty} \mu^*(I_j) < \mu^*(A) + \frac{\epsilon}{2}.$$

Para cada $j \in \mathbb{N}$, podemos elegir un intervalo acotado abierto $E_j(t_j)$ con $I_j \subset E_j(t_j)$ que cumpla que $\mu^*(E_j(t_j)) < \mu^*(I_j) + 2^{-(j+1)}\epsilon$, tal como se discutió en [\(1.4\)](#). Tomando el abierto $\tilde{E} = \bigcup_{j=1}^{\infty} E_j(t_j)$, que claramente es tal que $A \subset \tilde{E}$, se tiene

$$\mathcal{F} \leq \mu^*(\tilde{E}) \leq \sum_{j=1}^{\infty} \mu^*(E_j(t_j)) < \sum_{j=1}^{\infty} \mu^*(I_j) + \frac{\epsilon}{2} < \mu^*(A) + \epsilon.$$

Como ϵ es arbitrario, tenemos la desigualdad buscada: $\mathcal{F} \leq \mu^*(A)$. \square

1.2.4. Medida exterior nula e infinita

Existen conjuntos cuya medida exterior es infinita, que serán aquellos que necesitan una cantidad infinita numerable de intervalos acotados de medida elemental no nula para ser cubiertos. Por ejemplo, por la monotonía de μ^* , tenemos que $(2k)^n = \mu^*([-k, k]^n) \leq \mu^*(\mathbb{R}^n) \forall k \in \mathbb{N}$ y tomando límite, es claro que $\mu^*(\mathbb{R}^n) = \infty$.

La medida exterior de un conjunto también puede ser cero, si existe un recubrimiento numerable del mismo por intervalos de medida elemental nula. Presentamos algunos ejemplos de estos tipos de conjuntos.

Proposición 1.2.3. *Una unión numerable de conjuntos de medida exterior nula también tiene medida exterior nula.*

Demostración. Sea $\{A_j\} \subset \mathcal{P}(\mathbb{R}^n)$, con $\mu^*(A_j) = 0 \forall j \in \mathbb{N}$, por σ -subaditividad, se tiene que $0 \leq \mu^*(\bigcup_{j=1}^{\infty} A_j) \leq \sum_{j=1}^{\infty} \mu^*(A_j) = 0$. \square

Corolario 1.2.3.1. *Los conjuntos numerables de \mathbb{R}^n , como \mathbb{N} , \mathbb{Z} o \mathbb{Q} , tienen medida exterior nula; ya que para cualquier $x \in \mathbb{R}^n$, se cumple que $\mu^*(\{x\}) = 0$, al ser $\{x\}$ un intervalo (acotado) degenerado.*

Proposición 1.2.4. *Sean $d_1, d_2 \in \mathbb{N}$, y conjuntos $A \subset \mathbb{R}^{d_1}$, $B \subset \mathbb{R}^{d_2}$, tales que al menos una de $\mu_{d_1}^*(A)$ ó $\mu_{d_2}^*(B)$ es nula o infinita. Entonces*

$$\mu_{d_1+d_2}^*(A \times B) = \mu_{d_1}^*(A) \cdot \mu_{d_2}^*(B),$$

conviniendo que $\infty \cdot 0 = 0 \cdot \infty = 0$, $\infty \cdot c = c \cdot \infty = \infty \quad \forall c \in \mathbb{R}^+$ y $\infty \cdot \infty = \infty$. En general, se verifica $\mu_{d_1+d_2}^*(A \times B) \leq \mu_{d_1}^*(A) \cdot \mu_{d_2}^*(B) \forall A \subset \mathbb{R}^{d_1}, \forall B \subset \mathbb{R}^{d_2}$.

Demostración. Notamos $I_j = A_j \times B_j$ con $I_j \in \mathcal{I}_{d_1+d_2}$, $A_j \in \mathcal{I}_{d_1}$ y $B_j \in \mathcal{I}_{d_2}$ para cada $j \in \mathbb{N}$. Vemos que si $\{I_j\} \subset \mathcal{I}_{d_1+d_2}$ es un recubrimiento arbitrario de $A \times B$ por intervalos acotados de $\mathbb{R}^{d_1+d_2}$, entonces $\{A_j\} \subset \mathcal{I}_{d_1}$ recubre a A y $\{B_j\} \subset \mathcal{I}_{d_2}$ recubre a B . De ser estos dos últimos recubrimientos arbitrarios, $\{I_j\}$ también recubre a $A \times B$. Ahora, teniendo en cuenta que $\mu_{d_1+d_2}^*(A_j \times B_j) = \mu_{d_1}^*(A_j) \cdot \mu_{d_2}^*(B_j) \forall j \in \mathbb{N}$ como consecuencia del [Proposición 1.2.2](#),

$$\mu_{d_1+d_2}^*(A \times B) \leq \sum_{j=1}^{\infty} \mu_{d_1+d_2}^*(I_j) = \sum_{j=1}^{\infty} \mu_{d_1}^*(A_j) \cdot \sum_{j=1}^{\infty} \mu_{d_2}^*(B_j). \quad (1.5)$$

A continuación, puntuizamos los casos en los que se da la igualdad del enunciado:

- Si $\mu_{d_1}^*(A) = 0$, dado un $\epsilon > 0$, podemos escoger $\{A_j\}$ tal que $\sum_{j=1}^{\infty} \mu_{d_1}^*(A_j) < \mu_{d_1}^*(A) + \epsilon = \epsilon$. Por tanto, por la desigualdad (1.5),

$$\mu_{d_1+d_2}^*(A \times B) < \epsilon \cdot \sum_{j=1}^{\infty} \mu_{d_2}^*(B_j) = \sum_{j=1}^{\infty} \epsilon \cdot \mu_{d_2}^*(B_j) = \sum_{j=1}^{\infty} \mu_{d_2+1}^*([0, \epsilon] \times B_j).$$

Cuando $\epsilon \rightarrow 0$, notamos que $0 \leq \mu_{d_1+d_2}^*(A \times B) \leq \sum_{j=1}^{\infty} \mu_{d_2+1}^*([0, \epsilon] \times B_j) = 0$, por ser $\{0\} \times B_j \subset \mathbb{R}^{d_2+1}$ un intervalo acotado degenerado para todo $j \in \mathbb{N}$, que tiene por tanto medida $\mu_{d_2+1}^*$ nula. El procedimiento es análogo si $\mu_{d_2}^*(B) = 0$.

- Suponiendo $\mu_{d_1}^*(A) = \infty$ y $\mu_{d_2}^*(B)$ no nula, $\sum_{j=1}^{\infty} \mu_{d_1}^*(A_j) = \infty$ para cualquier recubrimiento $\{A_j\}$ de A , por lo que la suma $\sum_{j=1}^{\infty} \mu_{d_1+d_2}^*(I_j)$ en (1.5) ha de ser infinita para

cualquier familia $\{I_j\}$ que recubra a $A \times B$, lo cual nos dice que $\mu_{d_1+d_2}^*(A \times B) = \infty$. En caso de que $\mu_{d_2}^*(B) = \infty$ y $\mu_{d_1}^*(A) \neq 0$, el procedimiento es análogo.

Habiendo distinguido estos casos, y usando el convenio de notación del enunciado, basta tomar $\{A_j\}$ y $\{B_j\}$ arbitrarios en (1.5) para confirmar que $\mu_{d_1+d_2}^*(A \times B) \leq \mu_{d_1}^*(A) \cdot \mu_{d_2}^*(B)$. \square

Corolario 1.2.4.1. Los hiperplanos afines de \mathbb{R}^n se pueden expresar como productos cartesianos, siendo una de sus proyecciones un punto en \mathbb{R} , y por tanto tendrán medida exterior nula. En particular, cualquier conjunto contenido en uno de ellos también tendrá medida exterior nula, por la monotonía de μ^* .

1.3. Conjuntos medibles de \mathbb{R}^n

Definición 1.3.1. Se dice que un conjunto $A \subset \mathbb{R}^n$ es **medible** si cumple que

$$\mu^*(X) = \mu^*(X \cap A) + \mu^*(X \setminus A) \quad \forall X \subset \mathbb{R}^n.$$

De hecho, sólo será necesario que el conjunto cumpla la desigualdad

$$\mu^*(X \cap A) + \mu^*(X \setminus A) \leq \mu^*(X) \quad \forall X \subset \mathbb{R}^n \quad (1.6)$$

para ser medible, puesto que la otra ya se da por la σ -subaditividad de μ^* .

Llamamos \mathcal{M} a la familia de todos los conjuntos medibles de \mathbb{R}^n . En contextos donde se puede generar confusión, la denotaremos por \mathcal{M}_n .

1.3.1. σ -álgebra de los conjuntos medibles

Propiedad 1.3.1. \mathbb{R}^n es un conjunto medible.

Demostración. Se cumple que $\mu^*(X) = \mu^*(X) + \mu^*(\emptyset) = \mu^*(X \cap \mathbb{R}^n) + \mu^*(X \setminus \mathbb{R}^n) \quad \forall X \subset \mathbb{R}^n$, luego $\mathbb{R}^n \in \mathcal{M}$. \square

Propiedad 1.3.2. Si $A \subset \mathbb{R}^n$ es medible, su complementario $A^c = \mathbb{R}^n \setminus A$ también lo es.

Demostración. Sea $A \in \mathcal{M}$, vemos que $\mu^*(X \cap A^c) + \mu^*(X \setminus A^c) = \mu^*(X \setminus A) + \mu^*(X \cap A) = \mu^*(X) \quad \forall X \subset \mathbb{R}^n$, por lo que $A^c \in \mathcal{M}$. \square

Propiedad 1.3.3. Sea una secuencia finita de conjuntos medibles $A_1, A_2, \dots, A_m \subset \mathbb{R}^n$, $m \in \mathbb{N}$. Entonces, su unión $\bigcup_{j=1}^m A_j$ es medible.

Demostración. Sea $X \subset \mathbb{R}^n$ arbitrario. Como A_2 es medible se tiene que

$$\mu^*(X \setminus A_1) = \mu^*((X \setminus A_1) \cap A_2) + \mu^*((X \setminus A_1) \setminus A_2).$$

Notamos que $X \cap (A_1 \cup A_2) = (X \cap A_1) \cup ((X \setminus A_1) \cap A_2)$. Usando estas observaciones, la σ -subaditividad de μ^* y que A_1 es medible, tenemos

$$\begin{aligned} & \mu^*(X \cap (A_1 \cup A_2)) + \mu^*(X \setminus (A_1 \cup A_2)) \\ & \leq \mu^*(X \cap A_1) + \mu^*((X \setminus A_1) \cap A_2) + \mu^*((X \setminus A_1) \setminus A_2) \\ & = \mu^*(X \cap A_1) + \mu^*(X \setminus A_1) = \mu^*(X). \end{aligned}$$

Esto prueba que $A_1 \cup A_2 \in \mathcal{M}$ por (1.6). Aplicando inducción sobre m , obtenemos que $\bigcup_{j=1}^m A_j \in \mathcal{M}$. \square

Propiedad 1.3.4. *Sea $\{A_j\} \subset \mathcal{M}$ una familia numerable de conjuntos medibles, su unión $\bigcup_{j=1}^{\infty} A_j$ también es medible.*

Demostración. Dado $j \in \mathbb{N}$, definimos el conjunto

$$B_j = A_j \setminus \left(\bigcup_{k=1}^{j-1} A_k \right). \quad (1.7)$$

Por definición, $B_{j_1} \cap B_{j_2} = \emptyset \forall j_1, j_2 \in \mathbb{N}, j_1 \neq j_2$. Además, para cualquier $m \in \mathbb{N}$, la unión finita disjunta $S_m = \bigcup_{j=1}^m B_j$ es igual a $\bigcup_{j=1}^m A_j$, que es medible por ser unión finita de medibles, luego $S_m \in \mathcal{M}$. En particular, como $S_1 \in \mathcal{M}$, notamos que

$$\mu^*(X \cap S_2) = \mu^*((X \cap S_2) \cap S_1) + \mu^*((X \cap S_2) \setminus S_1) = \mu^*(X \cap S_1) + \mu^*(X \cap B_2) \quad \forall X \subset \mathbb{R}^n.$$

Aplicando el proceso inductivo sobre m , nos queda que

$$\mu^*(X \cap S_m) = \mu^*(X \cap S_{m-1}) + \mu^*(X \cap B_m) = \sum_{j=1}^m \mu^*(X \cap B_j) \quad \forall X \subset \mathbb{R}^n. \quad (1.8)$$

Sea $S_{\infty} = \bigcup_{j=1}^{\infty} B_j$, se cumple que $S_m \subset S_{\infty} \forall m \in \mathbb{N}$, luego dado $X \subset \mathbb{R}^n$, $\mu^*(X \setminus S_{\infty}) \leq \mu^*(X \setminus S_m) \forall m \in \mathbb{N}$, por la monotonía de μ^* . Usando esto, la medibilidad de S_m para cada $m \in \mathbb{N}$ y (1.8), es cierto que

$$\sum_{j=1}^m \mu^*(X \cap B_j) + \mu^*(X \setminus S_{\infty}) \leq \mu^*(X \cap S_m) + \mu^*(X \setminus S_m) = \mu^*(X) \quad \forall X \subset \mathbb{R}^n, \forall m \in \mathbb{N}.$$

Tomando límite, $\sum_{j=1}^{\infty} \mu^*(X \cap B_j) + \mu^*(X \setminus S_{\infty}) \leq \mu^*(X) \quad \forall X \subset \mathbb{R}^n$. También vemos que por σ -subaditividad, $\mu^*(X \cap S_{\infty}) = \mu^*(\bigcup_{j=1}^{\infty} (X \cap B_j)) \leq \sum_{j=1}^{\infty} \mu^*(X \cap B_j)$. Por las dos últimas desigualdades,

$$\mu^*(X) \leq \mu^*(X \cap S_{\infty}) + \mu^*(X \setminus S_{\infty}) \leq \sum_{j=1}^{\infty} \mu^*(X \cap B_j) + \mu^*(X \setminus S_{\infty}) \leq \mu^*(X) \quad \forall X \subset \mathbb{R}^n. \quad (1.9)$$

Se concluye pues que $S_{\infty} = \bigcup_{j=1}^{\infty} B_j = \bigcup_{j=1}^{\infty} A_j$ es medible. \square

Observación 1.3.1. Hemos probado pues que la familia \mathcal{M} contiene al total, es cerrada bajo complementarios y es cerrada bajo uniones numerables. Estas tres propiedades se pueden resumir diciendo que \mathcal{M} es un σ -álgebra en \mathbb{R}^n .

Observación 1.3.2. Cualquier otra operación que se pueda aplicar sobre una familia finita o infinita de conjuntos medibles y que involucre complementarios y uniones finitas o infinitas, también generará un conjunto medible. Por ejemplo, si $A_1, A_2, \dots, A_m \in \mathcal{M}$, $m \in \mathbb{N}$, la intersección finita $\bigcap_{j=1}^m A_j$ es medible, pues cada A_j^c con $j \in \Gamma_m$ es medible, luego $\bigcup_{j=1}^m (A_j^c) = (\bigcap_{j=1}^m A_j)^c \in \mathcal{M}$, y por ende su complementario $\bigcap_{j=1}^m A_j \in \mathcal{M}$. Análogamente, se puede comprobar el caso de las intersecciones infinitas. Así mismo, operaciones como la diferencia (\setminus) o la diferencia simétrica (Δ) se pueden expresar en estos términos.

1.3.2. Medibilidad de los intervalos y los abiertos de \mathbb{R}^n

Proposición 1.3.1. *Todo intervalo acotado de \mathbb{R}^n es un conjunto medible.*

Demostración. Sea $i \in \Gamma_n$ fijo y $S \subset \mathbb{R}$ una semirecta, definimos $A = \pi_i^{-1}(S)$, es decir, el conjunto de puntos cuya i -ésima componente está en la semirecta S . Para $X \subset \mathbb{R}^n$ y $\epsilon > 0$ arbitrarios, podemos encontrar un recubrimiento $\{I_j\} \subset \mathcal{I}$ tal que $\sum_{j=1}^{\infty} \mu^*(I_j) < \mu^*(X) + \epsilon$. Para cada $j \in \mathbb{N}$, los conjuntos

$$I_j \cap A = \{x \in I_j : \pi_i(x) \in S\} \quad \text{y} \quad I_j \setminus A = \{x \in I_j : \pi_i(x) \in \mathbb{R} \setminus S\}$$

son dos intervalos acotados de \mathbb{R}^n , disjuntos y cuya unión es exactamente I_j . Por ello, $\mu^*(I_j) = \mu^*(I_j \cap A) + \mu^*(I_j \setminus A)$, usando [Proposición 1.2.1](#). Ahora bien,

$$\mu^*(X \cap A) + \mu^*(X \setminus A) \leq \sum_{j=1}^{\infty} \mu^*(I_j \cap A) + \sum_{j=1}^{\infty} \mu^*(I_j \setminus A) = \sum_{j=1}^{\infty} \mu^*(I_j) < \mu^*(X) + \epsilon.$$

La arbitrariedad de ϵ hace que se cumpla la desigualdad ([1.6](#)) para un $X \subset \mathbb{R}^n$ cualquiera, luego podemos decir que los conjuntos de la forma $\pi_i^{-1}(S)$ con $S \subset \mathbb{R}$ una semirecta e $i \in \Gamma_n$ son medibles.

Sea $I \subset \mathcal{I}$ un intervalo acotado de \mathbb{R}^n , y sea $i \in \Gamma_n$. Podemos reescribir cualquier intervalo proyectando $\pi_i(I) \subset \mathbb{R}$ como intersección de semirectas $R_i, L_i \subset \mathbb{R}$, donde el extremo de cada una de ellas coincide con uno de los extremos de $\pi_i(I)$, siendo cerrados o abiertos según el caso. Esto es, $\pi_i(I) = R_i \cap L_i \quad \forall i \in \Gamma_n$. En consecuencia, tenemos que

$$I = \{x \in \mathbb{R}^n : \pi_i(x) \in R_i \cap L_i \quad \forall i \in \Gamma_n\} = \bigcap_{i=1}^n (\pi_i^{-1}(R_i) \cap \pi_i^{-1}(L_i)).$$

Como I es una intersección finita de conjuntos medibles, tenemos que $I \in \mathcal{M}$. □

Proposición 1.3.2. *Los conjuntos abiertos y los conjuntos cerrados de \mathbb{R}^n son medibles.*

Demostración. Esto es consecuencia directa de que los intervalos acotados de \mathbb{R}^n son medibles según la [Proposición 1.3.1](#) y todo abierto es unión numerable de cubos diádicos (disjuntos) según la [Proposición 1.1.3](#). Como \mathcal{M} es cerrada por uniones numerables, cualquier abierto ha de ser medible. Además, como los cerrados son los complementarios de los abiertos, también han de estar en \mathcal{M} . □

1.3.3. Caracterización de la medibilidad

Proposición 1.3.3. *Todo conjunto de medida exterior nula es medible.*

Demostración. Dado $A \subset \mathbb{R}^n$ con $\mu^*(A) = 0$, para cualquier $X \subset \mathbb{R}^n$ se cumple que

$$\mu^*(X \cap A) \leq \mu^*(A) = 0 \quad \text{y} \quad \mu^*(X \setminus A) \leq \mu^*(X)$$

usando la monotonía de μ^* . Por tanto, $\mu^*(X \cap A) + \mu^*(X \setminus A) \leq \mu^*(X)$, lo cual implica que $A \in \mathcal{M}$ por ([1.6](#)). □

Proposición 1.3.4. Dado un conjunto $A \subset \mathbb{R}^n$,

$$\begin{aligned} A \in \mathcal{M} &\iff \forall \epsilon > 0 \exists E \text{ abierto de } \mathbb{R}^n \text{ tal que } A \subset E \text{ y } \mu^*(E \setminus A) < \epsilon \iff \\ &\iff \exists G \subset \mathbb{R}^n, \text{ de tipo } G_\delta, \text{ tal que } A \subset G \text{ y } \mu^*(G \setminus A) = 0 \end{aligned}$$

Demostración. Dado $A \in \mathcal{M}$ y $k \in \mathbb{N}$, podemos tomar $A_k = A \cap [-k, k]^n$, que es medible por ser intersección de medibles. Entonces $\mu^*(A_k) < \infty$, pues A_k está contenido en el intervalo acotado $[-k, k]^n$. Dado $\epsilon > 0$, por la regularidad exterior de μ^* , podemos tomar un $E_k \subset \mathbb{R}^n$ abierto que contiene a A_k de forma que $\mu^*(E_k) < \mu^*(A_k) + 2^{-k}\epsilon$. Como $A_k \in \mathcal{M}$, entonces $\mu^*(E_k \setminus A_k) + \mu^*(A_k) = \mu^*(E_k)$, por lo que es cierto que $\mu^*(E_k \setminus A_k) < 2^{-k}\epsilon$. Considerando el abierto $E = \bigcup_{k=1}^{\infty} E_k$, por σ -subaditividad, se verifica que

$$\mu^*(E \setminus A) \leq \sum_{k=1}^{\infty} \mu^*(E_k \setminus A_k) < \sum_{k=1}^{\infty} \frac{\epsilon}{2^k} = \epsilon.$$

Hemos probado que la primera afirmación implica la segunda. En condiciones de la segunda, podemos tomar abiertos $\{G_j\}$ tal que $A \subset G_j$ y $\mu^*(G_j \setminus A) < \frac{1}{j}$ para cada $j \in \mathbb{N}$. Por tanto, la intersección $G = \bigcap_{j=1}^{\infty} G_j$, que es un conjunto de tipo G_δ , contiene a A y verifica que $\mu^*(G \setminus A) \leq \mu^*(G_j \setminus A) < \frac{1}{j} \forall j \in \mathbb{N}$, lo cual implica que $\mu^*(G \setminus A) = 0$.

Por último, notamos que si se cumple la tercera afirmación, A es medible, ya que $G \setminus A$ será medible por tener medida nula segú la [Proposición 1.3.3](#), y G es medible por ser intersección de abiertos (que son medibles), luego como $A \subset G$, $A = G \setminus (G \setminus A) \in \mathcal{M}$. \square

1.4. Medida de Lebesgue

Definición 1.4.1. Definimos la **medida de Lebesgue** como una función $\mu : \mathcal{M} \rightarrow \mathbb{R}_0^+ \cup \{\infty\}$ que es, exactamente, la restricción de la medida exterior μ^* a la familia \mathcal{M} de los conjuntos medibles. Esto es, $\mu = \mu^*|_{\mathcal{M}}$. En contextos ambiguos, la notaremos por μ_n .

Se hace necesario considerar esta restricción puesto que es posible definir subconjuntos de \mathbb{R}^n que no cumplen la propiedad de medibilidad, lo cual genera comportamientos no deseables en la medida exterior. A partir de ahora, se usará la medida μ en lugar de μ^* para todo conjunto que hemos demostrado que es medible, notando además que μ hereda todas las propiedades que ya vimos para μ^* .

1.4.1. Propiedades fundamentales

Definición 1.4.2. Una familia de conjuntos medibles $\{A_j\} \subset \mathcal{M}$ es **casi disjunta** si se puede expresar como una familia disjunta excepto, a lo sumo, en un conjunto de puntos de medida nula. Es decir, la unión de las intersecciones dos a dos entre conjuntos de la familia es tal que:

$$\mu\left(\bigcup_{j=1}^{\infty} \bigcup_{k=1}^{\infty} (A_j \cap A_k)\right) = 0.$$

Por el crecimiento y la σ -subaditividad de μ , esto equivale $\mu(A_j \cap A_k) = 0 \forall j, k \in \mathbb{N}$

Propiedad 1.4.1 (σ -aditividad). Si $\{A_j\} \subset \mathcal{M}$ es una familia numerable casi disjunta de conjuntos medibles de \mathbb{R}^n , entonces

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

Demostración. Para cada $j \in \mathbb{N}$, definimos el conjunto B_j de la forma (1.7). Tomando $S_{\infty} = \bigsqcup_{j=1}^{\infty} B_j = \bigcup_{j=1}^{\infty} A_j$, y aplicando $X = S_{\infty}$ sobre la igualdad en (1.9), se obtiene:

$$\mu(S_{\infty}) = \sum_{j=1}^{\infty} \mu(B_j). \quad (1.10)$$

Dado un $j \in \mathbb{N}$, podemos expresar

$$A_j = \left(A_j \setminus \left(\bigcup_{k=1}^{j-1} A_k \right) \right) \cup \left(A_j \cap \left(\bigcup_{k=1}^{j-1} A_k \right) \right) = B_j \cup \left(\bigcup_{k=1}^{j-1} A_j \cap A_k \right)$$

y por ser $\{A_j\}$ casi disjunta, es cierto que

$$\mu(A_j) \leq \mu(B_j) + \mu\left(\bigcup_{k=1}^{j-1} A_j \cap A_k\right) \leq \mu(B_j) + \sum_{k=1}^{j-1} \mu(A_j \cap A_k) = \mu(B_j).$$

La otra desigualdad se deriva de que por definición, $B_j \subset A_j$, luego $\mu(B_j) \leq \mu(A_j)$. Se tiene pues que $\mu(A_j) = \mu(B_j) \forall j \in \mathbb{N}$. Sustituyendo en (1.10) para cada $j \in \mathbb{N}$, obtenemos la igualdad deseada. \square

Corolario 1.4.1.1. *La aditividad finita se verifica también, por el mismo argumento utilizado en la Observación 1.2.1.*

Corolario 1.4.1.2. *Si $A, B \in \mathcal{M}$ con $B \subset A$, se tiene que $\mu(A \setminus B) = 0 \iff \mu(A) = \mu(B)$. Similarmente, $\mu(B) = 0 \iff \mu(A \setminus B) = \mu(A)$.*

Demostración. Expresamos $A = B \sqcup (A \setminus B)$ como unión disjunta, con lo que $\mu(A) = \mu(B \sqcup (A \setminus B)) = \mu(B) + \mu(A \setminus B)$. La nulidad de uno de los términos determina la medida de A , y la igualdad de $\mu(A)$ con alguno de los términos implica la nulidad del otro. \square

Propiedad 1.4.2 (Continuidad creciente). *Si $\{A_j\} \subset \mathcal{M}$ es una familia de conjuntos medibles de \mathbb{R}^n tal que $A_j \subset A_{j+1}$ para cada $j \in \mathbb{N}$ y $A = \bigcup_{j=1}^{\infty} A_j$, entonces $\lim_{j \rightarrow \infty} \mu(A_j) = \mu(A)$.*

Demostración. Tomando $A_0 = \emptyset$ y $B_j = A_j \setminus A_{j-1}$ para cada $j \in \mathbb{N}$, tenemos las uniones disjuntas $A_j = \bigsqcup_{k=1}^j B_k \forall j \in \mathbb{N}$ y $A = \bigsqcup_{j=1}^{\infty} B_j$, luego por la σ -aditividad,

$$\lim_{j \rightarrow \infty} \mu(A_j) = \lim_{j \rightarrow \infty} \mu\left(\bigsqcup_{k=1}^j B_k\right) = \lim_{j \rightarrow \infty} \sum_{k=1}^j \mu(B_k) = \mu\left(\bigsqcup_{j=1}^{\infty} B_j\right) = \mu(A).$$

\square

1.4.2. Propiedades geométricas

Propiedad 1.4.3 (Invariancia por traslaciones). *Sea $A \subset \mathbb{R}^n$ medible y $x \in \mathbb{R}^n$. Entonces,*

$$x + A \in \mathcal{M} \quad y \quad \mu(A) = \mu(x + A).$$

Demostración. Sea $x \in \mathbb{R}^n$, un conjunto $B \subset \mathbb{R}^n$ y un recubrimiento $\{I_j\} \subset \mathcal{I}$ de B . Entonces, $\{x + I_j\} \subset \mathcal{I}$ recubre al conjunto $x + B$. Además, para cada $j \in \mathbb{N}$, las proyecciones del intervalo $x + I_j$ tienen la misma longitud que las del intervalo I_j . Por tanto, $\mu(I_j) = \mu(x + I_j)$, y $\mu^*(x + B) \leq \sum_{j=1}^{\infty} \mu(x + I_j) = \sum_{j=1}^{\infty} \mu(I_j)$. La arbitrariedad de la familia $\{I_j\} \subset \mathcal{I}$ nos da la desigualdad $\mu^*(x + B) \leq \mu^*(B)$. Si trasladamos el conjunto $B' = x + B$ por el punto $y = -x$, tenemos que $\mu^*(B) = \mu^*(y + B') \leq \mu^*(B') = \mu^*(x + B)$, lo cual confirma que

$$\mu^*(B) = \mu^*(x + B) \quad \forall B \subset \mathbb{R}^n, \forall x \in \mathbb{R}^n. \quad (1.11)$$

En particular, la igualdad es válida para conjuntos medibles. Dados $x \in \mathbb{R}^n$ y $A \in \mathcal{M}$, para $\epsilon > 0$, podemos encontrar un abierto E tal que $A \subset E$ y $\mu(E \setminus A) < \epsilon$ usando la **Proposición 1.3.4**. En tal caso $x + E$ es abierto y $x + A \subset x + E$ de forma que $(x + E) \setminus (x + A) = x + E \setminus A$, luego tomando $B = E \setminus A$ en (1.11),

$$\mu^*((x + E) \setminus (x + A)) = \mu^*(x + E \setminus A) = \mu(E \setminus A) < \epsilon.$$

La arbitrariedad de ϵ demuestra que $x + A \in \mathcal{M}$. \square

Propiedad 1.4.4 (Homogeneidad). *Para cada $j \in \Gamma_n$ fijo y cada $\rho \in \mathbb{R}$, sea la aplicación*

$$\begin{aligned} \phi_{\rho}^{(j)} : \mathbb{R}^n &\rightarrow \mathbb{R}^n \\ x &\mapsto (x_1, x_2, \dots, \rho \cdot x_j, \dots, x_n). \end{aligned} \quad (1.12)$$

Si $A \in \mathcal{M}$, entonces $\phi_{\rho}^{(j)}(A) \in \mathcal{M}$ verificando que $\mu(\phi_{\rho}^{(j)}(A)) = |\rho| \cdot \mu(A)$.

Demostración. Si $\rho = 0$ y $A \in \mathcal{M}$, $\phi_{\rho}^{(j)}(A)$ está contenido en el hiperplano afín $\pi_j^{-1}\{0\}$, por lo que $\mu^*(\phi_{\rho}^{(j)}(A)) = 0$ por el **Corolario 1.2.4.1**.

Por otro lado, fijando $j \in \Gamma_n$ y $\rho \in \mathbb{R} \setminus \{0\}$, tomamos $\phi := \phi_{\rho}^{(j)}$. Tanto ϕ como $\phi^{-1} = \phi_{1/\rho}^{(j)}$ son continuas y biyectivas, luego ϕ es un homeomorfismo.

Como podemos expresar cualquier abierto $E \subset \mathbb{R}^n$ como $E = \bigcup_{j=1}^{\infty} D_j$, donde $\{D_j\}$ es una familia de cubos diádicos disjuntos, se tiene que $\phi(E) = \bigcup_{j=1}^{\infty} \phi(D_j)$ es una unión disjunta por la biyectividad de ϕ . Además, $\mu(\phi(D_j)) = |\rho| \cdot \mu(D_j)$ para cada $j \in \mathbb{N}$, pues ϕ escala la j -ésima proyección de D_j por el factor ρ , quedando un intervalo con longitud escalada por ρ en dicha proyección. Entonces, por la σ -subaditividad de μ ,

$$\mu(\phi(E)) = \sum_{j=1}^{\infty} \mu(\phi(D_j)) = \sum_{j=1}^{\infty} |\rho| \cdot \mu(D_j) = |\rho| \cdot \sum_{j=1}^{\infty} \mu(D_j) = |\rho| \cdot \mu(E).$$

Sean $B, G \subset \mathbb{R}^n$, con G abierto y $\phi(B) \subset G$, sabemos que existe un E abierto de \mathbb{R}^n con $\phi(E) = G$, por ser ϕ homeomorfismo. Además, $\phi(B) \subset \phi(E)$ si y solo si $B \subset E$.

Por tanto, por la regularidad exterior de μ^* ,

$$\begin{aligned} \mu^*(\phi(B)) &= \inf \{\mu(\phi(E)) : B \subset E, E \text{ abierto de } \mathbb{R}^n\} = \\ &= \inf \{|\rho| \cdot \mu(E) : B \subset E, E \text{ abierto de } \mathbb{R}^n\} = |\rho| \cdot \mu^*(B) \quad \forall B \subset \mathbb{R}^n. \end{aligned}$$

Si $A \in \mathcal{M}$, para cada $\epsilon > 0$, encontramos $E \subset \mathbb{R}^n$ abierto tal que $A \subset E$ y $\mu(E \setminus A) < \frac{\epsilon}{|\rho|}$, luego por ser ϕ homeomorfismo, $\mu^*(\phi(E) \setminus \phi(A)) = \mu^*(\phi(E \setminus A)) = |\rho| \cdot \mu(E \setminus A) < \epsilon$, donde $\phi(E)$ es abierto con $\phi(A) \subset \phi(E)$. Por tanto, $\phi(A) \in \mathcal{M}$ y $\mu(\phi(A)) = |\rho| \cdot \mu(A)$. \square

Corolario 1.4.4.1. Si $A \in \mathcal{M}$ y $\rho \in \mathbb{R}$, se cumple que $\mu(\rho \cdot A) = |\rho|^n \cdot \mu(A)$, pues $\rho \cdot A = \phi_\rho^{(1)} \circ \phi_\rho^{(2)} \circ \cdots \circ \phi_\rho^{(n)}(A)$.

Propiedad 1.4.5. Sean $d_1, d_2 \in \mathbb{N}$, A un conjunto medible en \mathbb{R}^{d_1} y B medible en \mathbb{R}^{d_2} . Entonces, el producto cartesiano $A \times B$ es medible en $\mathbb{R}^{d_1+d_2}$. Además,

$$\mu_{d_1+d_2}(A \times B) = \mu_{d_1}(A) \cdot \mu_{d_2}(B). \quad (1.13)$$

Demostración. Veamos en primer lugar que $A \times B \in \mathcal{M}_{d_1+d_2}$. Usando la Proposición 1.3.4, vemos que podemos tomar $G_1 \subset \mathbb{R}^{d_1}$ y $G_2 \subset \mathbb{R}^{d_2}$ tal que $A \subset G_1$ y $B \subset G_2$, ambos de tipo G_δ , verificando que $\mu_{d_1}^*(G_1 \setminus A) = 0$ y $\mu_{d_2}^*(G_2 \setminus B) = 0$. Se tiene que $A \times B \subset G_1 \times G_2 \subset \mathbb{R}^{d_1+d_2}$ y usando la identidad para la diferencia del producto cartesiano,

$$(G_1 \times G_2) \setminus (A \times B) = [(G_1 \setminus A) \times G_2] \cup [G_1 \times (G_2 \setminus B)] \subset \mathbb{R}^{d_1+d_2}.$$

Por la Proposición 1.2.4, $\mu_{d_1+d_2}^*[(G_1 \setminus A) \times G_2] = 0$ al ser $\mu_{d_1}^*(G_1 \setminus A) = 0$. Análogamente, $\mu_{d_1+d_2}^*[G_1 \times (G_2 \setminus B)] = 0$. Entonces, por σ -subaditividad,

$$\mu_{d_1+d_2}^*[(G_1 \times G_2) \setminus (A \times B)] = 0. \quad (1.14)$$

Obviamente, $G_1 \times G_2$ es un conjunto de tipo G_δ (intersección numerable de abiertos de $\mathbb{R}^{d_1+d_2}$), como producto cartesiano de intersecciones numerables de abiertos de \mathbb{R}^{d_1} y \mathbb{R}^{d_2} , respectivamente. Entonces, de nuevo por Proposición 1.3.4, tenemos que $A \times B \in \mathcal{M}_{d_1+d_2}$.

En caso de que una de $\mu_{d_1}(A)$ ó $\mu_{d_2}(B)$ sea nula o infinita, ya tenemos la igualdad (1.13), por lo visto en Proposición 1.2.4. En el caso de que ambas sean finitas, notamos que $G_1 = \bigcap_{j=1}^{\infty} E_{1,j}$ y $G_2 = \bigcap_{j=1}^{\infty} E_{2,j}$, donde $E_{1,j} \subset \mathbb{R}^{d_1}$ y $E_{2,j} \subset \mathbb{R}^{d_2}$ son abiertos para cada $j \in \mathbb{N}$. Ahora bien, $E_{1,j} = \bigsqcup_{k=1}^{\infty} D_{1,j,k}$ y $E_{2,j} = \bigsqcup_{k=1}^{\infty} D_{2,j,k}$, donde $D_{1,j,k} \in \mathcal{I}_{d_1}$ es un cubo diádico de \mathbb{R}^{d_1} y $D_{2,j,k} \in \mathcal{I}_{d_2}$ es un cubo diádico de \mathbb{R}^{d_2} , para todo $j, k \in \mathbb{N}$. Entonces,

$$G_1 \times G_2 = \bigcap_{j=1}^{\infty} (E_{1,j} \times E_{2,j}) = \bigsqcup_{k=1}^{\infty} \left(\bigcap_{j=1}^{\infty} (D_{1,j,k} \times D_{2,j,k}) \right) = \bigsqcup_{k=1}^{\infty} \left((\bigcap_{j=1}^{\infty} D_{1,j,k}) \times (\bigcap_{j=1}^{\infty} D_{2,j,k}) \right).$$

La intersección numerable de cubos diádicos es un intervalo acotado (pudiendo ser degenerado o vacío). Entonces, notando $I_{1,k} = \bigcap_{j=1}^{\infty} D_{1,j,k} \in \mathcal{I}_{d_1}$ y $I_{2,k} = \bigcap_{j=1}^{\infty} D_{2,j,k} \in \mathcal{I}_{d_2}$ para cada $k \in \mathbb{N}$, por la σ -aditividad de $\mu_{d_1+d_2}$ y la Proposición 1.2.2, vemos que

$$\mu_{d_1+d_2}(G_1 \times G_2) = \sum_{k=1}^{\infty} \mu_{d_1+d_2}(I_{1,k} \times I_{2,k}) = \sum_{k=1}^{\infty} \mu_{d_1}(I_{1,k}) \cdot \sum_{k=1}^{\infty} \mu_{d_2}(I_{2,k}) = \mu_{d_1}(G_1) \cdot \mu_{d_2}(G_2).$$

Como $A \in \mathcal{M}_{d_1}$, $B \in \mathcal{M}_{d_2}$ y $A \times B \in \mathcal{M}_{d_1+d_2}$, esta igualdad junto con (1.14) y Corolario 1.4.1.2, nos da que

$$\mu_{d_1+d_2}(A_1 \times A_2) = \mu_{d_1+d_2}(G_1 \times G_2) = \mu_{d_1}(G_1) \cdot \mu_{d_2}(G_2) = \mu_{d_1}(A) \cdot \mu_{d_2}(B).$$

□

1.5. Funciones reales medibles

Para esta sección y para la siguiente, fijaremos un $\Omega \subset \mathbb{R}^n$.

1.5.1. El espacio $\mathcal{L}(\Omega)$

Definición 1.5.1. Diremos que la función real $f : \Omega \rightarrow \mathbb{R}$ es **medible** si para cualquier abierto $E \subset \mathbb{R}$, se cumple que $f^{-1}(E)$ es medible en \mathbb{R}^n . En particular, $f^{-1}(\mathbb{R}) = \Omega \in \mathcal{M}$.

Notaremos por $\mathcal{L}(\Omega)$ al conjunto de todas las funciones reales medibles definidas en Ω , o simplemente \mathcal{L} si $\Omega = \mathbb{R}^n$.

Observación 1.5.1. Como cualquier intervalo $I \subset \mathbb{R}$ no abierto se puede aproximar como una intersección numerable de intervalos abiertos de \mathbb{R} , y la imagen inversa de una intersección es la intersección de imágenes inversas, tenemos que dado $f \in \mathcal{L}(\Omega)$, entonces $f^{-1}(I) \in \mathcal{M}$.

Observación 1.5.2. La funciones reales continuas definidas en Ω son medibles, pues para cualquier abierto $E \subset \mathbb{R}$, $f^{-1}(E) \subset \mathbb{R}^n$ es abierto, y por tanto medible.

Observación 1.5.3. Sea $f \in \mathcal{L}(\Omega)$ y $f|_A : A \rightarrow \mathbb{R}$ la restricción de f a un subdominio $A \subset \Omega$ medible. Dado E abierto de \mathbb{R} , $f|_A^{-1}(E) = f^{-1}(E) \cap A \in \mathcal{M}$ como intersección de medibles. Entonces, se tiene que $f|_A \in \mathcal{L}(A)$.

Proposición 1.5.1. Sea $k \in \mathbb{N}$ y funciones $f_1, f_2, \dots, f_k \in \mathcal{L}(\Omega)$. Cualquier transformación continua $\sigma : \mathbb{R}^k \rightarrow \mathbb{R}$ aplicada sobre ellas para cada punto de Ω resulta en una función medible.

Demostración. Definimos $\tau : \mathbb{R}^n \rightarrow \mathbb{R}^k$ con $\tau(x) = (f_1(x), f_2(x), \dots, f_k(x)) \forall x \in \mathbb{R}$. Sea E un abierto de \mathbb{R} , $\sigma^{-1}(E) = E'$ es un abierto de \mathbb{R}^k por ser σ continua, y se cumple que

$$\begin{aligned} x \in \tau^{-1}(E') &\iff (f_1(x), f_2(x), \dots, f_k(x)) \in E' \iff \\ &\iff f_i(x) \in \pi_i(E') \quad \forall i \in \Gamma_k \iff x \in \bigcap_{i=1}^k f_i^{-1}(\pi_i(E')). \end{aligned}$$

Sea $i \in \Gamma_k$, $\pi_i(E')$ es un abierto de \mathbb{R} , luego su imagen inversa por f_i es medible. La intersección de todas ellas, por tanto, también lo es. Esto implica que $\tau^{-1}(E') = (\sigma \circ \tau)^{-1}(E) \in \mathcal{M}$, luego $\sigma \circ \tau : \mathbb{R}^n \rightarrow \mathbb{R}$ es una función medible. \square

Corolario 1.5.1.1. Dado $k \in \mathbb{N}$, la suma sobre \mathbb{R}^k es una transformación continua, por lo que la suma de funciones medibles es medible. La multiplicación por escalares sobre \mathbb{R} también lo es, por lo que dado $\alpha \in \mathbb{R}$ y $f \in \mathcal{L}(\Omega)$, se cumple que $\alpha \cdot f \in \mathcal{L}(\Omega)$. Por tanto, $\mathcal{L}(\Omega)$ es un **subespacio vectorial** del espacio vectorial que forman las funciones reales definidas en Ω .

Observación 1.5.4. La **Proposición 1.5.1** también nos indica que $\mathcal{L}(\Omega)$ es cerrado bajo muchas otras operaciones algebraicas, como el producto, la potenciación, etc.

Proposición 1.5.2. Sea una sucesión $\{f_j\} \subset \mathcal{L}(\Omega)$ que converge puntualmente a la función $f : \Omega \rightarrow \mathbb{R}$. Entonces, $f \in \mathcal{L}(\Omega)$.

Demostración. Sea E un abierto de \mathbb{R} . Definimos el conjunto $E_k = \{x \in E : \text{dist}(x, E^c) > \frac{1}{k}\}$ para cada $k \in \mathbb{N}$. Dado $x \in E$, existe $k \in \mathbb{N}$ tal que la bola $B(x, \frac{1}{k}) \subset E$, por lo que $x \in E_k$.

Tomando $\epsilon < (\text{dist}(x, E^c) - \frac{1}{k})$, $B(x, \epsilon) \subset E_k$, luego E_k es abierto y $E = \bigcup_{k=1}^{\infty} E_k$. Entonces,

$$\begin{aligned} x \in f^{-1}(E) &\iff f(x) \in E_k \text{ para cierto } k \in \mathbb{N} \iff f_j(x) \in E_k \text{ para } j \geq j_0; j_0, k \in \mathbb{N} \iff \\ &\iff x \in \bigcap_{j=j_0}^{\infty} f_j^{-1}(E_k) \text{ para ciertos } j_0, k \in \mathbb{N} \iff x \in \bigcup_{k=0}^{\infty} \bigcup_{j=0}^{\infty} \bigcap_{j=j_0}^{\infty} f_j^{-1}(E_k). \end{aligned}$$

Entonces, $\bigcup_{k=0}^{\infty} \bigcup_{j_0=0}^{\infty} \bigcap_{j=j_0}^{\infty} f_j^{-1}(E_k) = f^{-1}(E) \in \mathcal{M}$ por ser unión de intersecciones de medibles. Al ser E un abierto arbitrario de \mathbb{R} , tenemos que $f \in \mathcal{L}(\Omega)$. \square

1.5.2. El conjunto $\mathcal{L}^+(\Omega)$

Definición 1.5.2. Si $f \in \mathcal{L}(\Omega)$ sólo toma valores en \mathbb{R}_0^+ , entonces diremos que f es una **función real medible positiva** en Ω . El conjunto de las funciones reales medibles positivas definidas en Ω se notará por $\mathcal{L}^+(\Omega)$, o \mathcal{L}^+ para $\Omega = \mathbb{R}^n$. Obviamente, $\mathcal{L}^+(\Omega) \subset \mathcal{L}(\Omega)$.

Definición 1.5.3. La **función característica** de un subconjunto $A \subset \mathbb{R}^n$ se define como:

$$\begin{aligned} \chi_A : \mathbb{R}^n &\rightarrow \mathbb{R} \\ x &\mapsto \begin{cases} 1 & \text{si } x \in A \\ 0 & \text{si } x \notin A. \end{cases} \end{aligned}$$

Proposición 1.5.3. Dado $A \subset \mathbb{R}^n$, A es medible si y solo si $\chi_A \in \mathcal{L}^+$.

Demostración. Sea E un abierto de \mathbb{R} , entonces sólo puede haber cuatro casos:

$$\chi_A^{-1}(E) = \begin{cases} \emptyset & \text{si } \{0,1\} \not\subset E \\ A^c & \text{si } 0 \in E, 1 \notin E \\ A & \text{si } 0 \notin E, 1 \in E \\ \mathbb{R}^n & \text{si } \{0,1\} \subset E. \end{cases}$$

Entonces, es evidente la equivalencia $A \in \mathcal{M} \iff \chi_A \in \mathcal{L}^+$. \square

Definición 1.5.4. Sea $A \subset \Omega$ y $f : \Omega \rightarrow \mathbb{R}$, llamamos **función característica** de f en A a la función $f \cdot \chi_A : \Omega \rightarrow \mathbb{R}$. Si $A \in \mathcal{M}$ y $f \in \mathcal{L}(\Omega)$, se tiene que $\chi_A \in \mathcal{L}(\Omega)$ como restricción de χ_A al conjunto medible Ω , luego $f \cdot \chi_A \in \mathcal{L}(\Omega)$ como producto de medibles.

Definición 1.5.5. Sea $f \in \mathcal{L}(\Omega)$. Consideraremos los conjuntos disjuntos

$$S_+ := f^{-1}([0, \infty[) \quad \text{y} \quad S_- := f^{-1}(]-\infty, 0[).$$

Los intervalos $]0, \infty[$ y $]-\infty, 0[$ son abiertos, luego $S_+, S_- \in \mathcal{M}$ por ser f medible. Definimos pues las funciones $f_+ := f \cdot \chi_{S_+} : \Omega \rightarrow \mathbb{R}$ y $f_- := -f \cdot \chi_{S_-} : \Omega \rightarrow \mathbb{R}$, que son ambas medibles. Estas funciones son las **partes positiva y negativa** de f , respectivamente.

Notemos que $f = f_+ - f_-$, con $f_+, f_- \in \mathcal{L}^+(\Omega)$. Esto es, cualquier función real medible puede expresarse en términos de funciones reales medibles positivas. Así mismo, podemos escribir el **valor absoluto** de f como $|f| = f_+ + f_-$, por lo que $|f| \in \mathcal{L}^+(\Omega)$.

1.6. Integral de Lebesgue

Existen diferentes alternativas para introducir el concepto de integral de Lebesgue de funciones reales medibles positivas. Usualmente, se suele definir en términos de funciones simples positivas, que son combinaciones lineales finitas de funciones características de conjuntos medibles. En nuestro estudio concreto, nos conviene presentar una definición con un enfoque más geométrico, que además nos permite realizar una interpretación más intuitiva del significado fundamental de la integración.

1.6.1. Gráficas y subgráficas

Definición 1.6.1. Sea una función $f : \Omega \rightarrow \mathbb{R}$, definimos la **gráfica de f** como el conjunto

$$\mathcal{G}(f) = \{(x, f(x)) \in \Omega \times \mathbb{R}\} \subset \mathbb{R}^{n+1}.$$

Proposición 1.6.1. Si $f \in \mathcal{L}(\Omega)$, la gráfica $\mathcal{G}(f)$ es un conjunto de medida nula en \mathbb{R}^{n+1} .

Demostración. Consideramos un recubrimiento arbitrario $\{I_j\} \subset \mathcal{I}_n$ de Ω .

Sea la unión disjunta $\mathbb{R}^+ = \bigsqcup_{k=1}^{\infty} [k-1, k] = \bigsqcup_{k=1}^{\infty} \bigsqcup_{l=1}^{m_k} J_{k,l}$, donde $m_k \in \mathbb{N}$ es arbitrario para cada $k \in \mathbb{N}$; y dados $k \in \mathbb{N}$ y $l \in \Gamma_{m_k}$, $J_{k,l} =]k - \frac{l-1}{m_k}, k - \frac{l}{m_k}]$. Fijando $j \in \mathbb{N}$, y sea el hiperplano $H = \mathbb{R}^n \times \{0\} \subset \mathbb{R}^{n+1}$, que cumple que $\mu_{n+1}(H) = 0$, se tiene

$$\mathcal{G}(f^+ \cdot \chi_{I_j}) \subset H \sqcup \left[(f^+ \cdot \chi_{I_j})^{-1}(\mathbb{R}^+) \times \mathbb{R}^+ \right] = H \sqcup \left[\bigsqcup_{k=1}^{\infty} \bigsqcup_{l=1}^{m_k} \left((f^+ \cdot \chi_{I_j})^{-1}(J_{k,l}) \times J_{k,l} \right) \right]. \quad (1.15)$$

Además, para cada $k \in \mathbb{N}$ y $l \in \Gamma_{m_k}$, $(f^+ \cdot \chi_{I_j})^{-1}(J_{k,l}) \in \mathcal{M}_n$ porque la función característica de f^+ en I_j es medible, al ser $f^+ \in \mathcal{L}(\Omega)$ e $I_j \in \mathcal{M}_n$, y usando la **Observación 1.5.1**. Además, por la **Propiedad 1.4.5**, su producto cartesiano con $J_{k,l}$ ha de ser medible en \mathbb{R}^{n+1} y

$$\begin{aligned} \mu_{n+1} \left((f^+ \cdot \chi_{I_j})^{-1}(J_{k,l}) \times J_{k,l} \right) &= \mu_n \left((f^+ \cdot \chi_{I_j})^{-1}(J_{k,l}) \right) \cdot \mu_1(J_{k,l}) \\ &= \frac{1}{m_k} \cdot \mu_n \left((f^+ \cdot \chi_{I_j})^{-1}(J_{k,l}) \right). \end{aligned}$$

Usando la inclusión (1.15), la σ -aditividad de μ_n y que $(f^+ \cdot \chi_{I_j})^{-1}(\mathbb{R}^+) \subset I_j \forall k \in \mathbb{N}$,

$$\begin{aligned} \mu_{n+1}^* [\mathcal{G}(f^+ \cdot \chi_{I_j})] &\leq \mu_{n+1}(H) + \sum_{k=1}^{\infty} \sum_{l=1}^{m_k} \frac{1}{m_k} \cdot \mu_n \left((f^+ \cdot \chi_{I_j})^{-1}(J_{k,l}) \right) \\ &= \sum_{k=1}^{\infty} \frac{1}{m_k} \cdot \mu_n \left(\bigsqcup_{l=1}^{m_k} ((f^+ \cdot \chi_{I_j})^{-1}(J_{k,l})) \right) \\ &= \sum_{k=1}^{\infty} \frac{1}{m_k} \cdot \mu_n \left((f^+ \cdot \chi_{I_j})^{-1}(\mathbb{R}^+) \right) \leq \sum_{k=1}^{\infty} \frac{1}{m_k} \cdot \mu_n(I_j). \end{aligned}$$

Dado $\epsilon > 0$, basta con tomar $m_k = \frac{2^k}{\epsilon}$ para cada $k \in \mathbb{N}$, de forma que $0 \leq \mu_{n+1}^* [\mathcal{G}(f^+ \cdot \chi_{I_j})] \leq \epsilon \cdot \mu_n(I_j)$. Como $\mu_n(I_j) < \infty$ por ser I_j intervalo acotado y ϵ arbitrario, concluimos que $\mu_{n+1}^* [\mathcal{G}(f^+ \cdot \chi_{I_j})] = 0$. Una discusión totalmente análoga nos da que $\mu_{n+1}^* [\mathcal{G}(f^- \cdot \chi_{I_j})] = 0$. Obviamente, $\mathcal{G}(f^+) \subset \bigcup_{j=1}^{\infty} \mathcal{G}(f^+ \cdot \chi_{I_j})$ y $\mathcal{G}(f^-) \subset \bigcup_{j=1}^{\infty} \mathcal{G}(f^- \cdot \chi_{I_j})$, pues $\{I_j\}$ recubre a Ω .

Por la [Proposición 1.2.3](#), $\mu_{n+1}[\mathcal{G}(f^+)] = 0$ y $\mu_{n+1}[\mathcal{G}(f^-)] = 0$. Véase que $\mathcal{G}(f) \subset \mathcal{G}(f^+) \cup \phi_{-1}^{n+1}(\mathcal{G}(f^-))$, donde ϕ_{-1}^{n+1} es la aplicación de [\(1.12\)](#) con $j = n + 1$ y $\rho = -1$. Concluimos pues que $\mu_{n+1}^*[\mathcal{G}(f)] = \mu_{n+1}[\mathcal{G}(f)] = 0$. \square

Definición 1.6.2. Sea $f : \Omega \rightarrow \mathbb{R}$ una función real que toma valores en \mathbb{R}_0^+ para todo Ω . Definimos la **subgráfica cerrada de f** como el conjunto

$$\mathcal{U}[f] = \{(x, y) \in \Omega \times \mathbb{R} : 0 \leq y \leq f(x)\} \subset \mathbb{R}^{n+1}.$$

Este conjunto representa la noción de espacio encerrado entre el hiperplano $\mathbb{R}^n \times \{0\}$ y la gráfica $\mathcal{G}(f)$, lo cual se puede interpretar como el espacio bajo la gráfica de f .

Análogamente, se puede definir la **subgráfica abierta $\mathcal{U}(f)$** tomando desigualdades estrictas en la definición del conjunto, la subgráfica cerrada por la derecha $\mathcal{U}(f)$ cuando sólo la primera desigualdad es estricta y la cerrada por la izquierda $\mathcal{U}[f]$ si sólo la segunda lo es.

Proposición 1.6.2. *Sea $f \in \mathcal{L}^+(\Omega)$, la subgráfica cerrada $\mathcal{U}[f]$ es medible.*

Demostración. Definimos la función $h : \Omega \times \mathbb{R} \rightarrow \mathbb{R}$ dada por $h(x, y) = f(x)$ para cada $(x, y) \in \Omega \times \mathbb{R}$. Dado un abierto $E \subset \mathbb{R}$, $h^{-1}(E) = f^{-1}(E) \times \mathbb{R} \in \mathcal{M}_{n+1}$ como producto cartesiano de medibles. Entonces, h es medible en $\Omega \times \mathbb{R}$. Además, la función $g := h - \pi_{n+1} : \Omega \times \mathbb{R} \rightarrow \mathbb{R}$ es también medible por la continuidad de la proyección y la [Proposición 1.5.1](#). Tomando el intervalo $[0, +\infty[\subset \mathbb{R}$, se tiene que $g^{-1}([0, +\infty[) = \{(x, y) \in \Omega \times \mathbb{R} : y \leq f(x)\} \in \mathcal{M}_{n+1}$. Claramente, $\Omega \times \mathbb{R}_0^+ \in \mathcal{M}_{n+1}$, y expresamos

$$\mathcal{U}[f] = \{(x, y) \in \Omega \times \mathbb{R} : 0 \leq y \leq f(x)\} = (\Omega \times \mathbb{R}_0^+) \cap g^{-1}([0, +\infty[).$$

Entonces, $\mathcal{U}[f] \in \mathcal{M}_{n+1}$ como intersección de medibles. \square

Corolario 1.6.2.1. *Dado $f \in \mathcal{L}^+(\Omega)$, las subgráficas $\mathcal{U}[f]$, $\mathcal{U}(f)$, $\mathcal{U}(f)$ y $\mathcal{U}[f]$ son todas medibles en \mathbb{R}^{n+1} y su medida μ_{n+1} es la misma.*

Demostración. Evidentemente, $\mathcal{U}[f] = \mathcal{G}(0) \sqcup \mathcal{U}(f) = \mathcal{U}[f] \sqcup \mathcal{G}(f) = \mathcal{G}(0) \sqcup \mathcal{U}(f) \sqcup \mathcal{G}(f)$. Las gráficas tienen medida nula por ser $0, f \in \mathcal{L}(\Omega)$. En consecuencia, son medibles, y aplicando diferencias entre conjuntos y la σ -aditividad de μ_{n+1} , es fácil ver que $\mathcal{U}[f]$, $\mathcal{U}(f)$, $\mathcal{U}(f)$, $\mathcal{U}[f] \in \mathcal{M}_{n+1}$ con

$$\mu_{n+1}[\mathcal{U}[f]] = \mu_{n+1}[\mathcal{U}(f)] = \mu_{n+1}[\mathcal{U}(f)] = \mu_{n+1}[\mathcal{U}[f]].$$

\square

Observación 1.6.1. Para el posterior desarrollo, únicamente nos interesarán medir el espacio bajo una función $f \in \mathcal{L}^+(\Omega)$, por lo que se puede utilizar cualquiera de las subgráficas introducidas, al tener medidas equivalentes. Nosotros escogeremos la subgráfica abierta $\mathcal{U}(f)$. Un importante detalle a notar es que, dado $A \subset \Omega$ medible, $f \cdot \chi_A \in \mathcal{L}^+(\Omega)$ y

$$\mathcal{U}(f \cdot \chi_A) = \{(x, y) \in \Omega \times \mathbb{R} : 0 < y < f \cdot \chi_A(x)\} = \{(x, y) \in A \times \mathbb{R} : 0 < y < f(x)\}.$$

Entonces, medir el espacio bajo la función $f \cdot \chi_A$ será equivalente a medir el espacio bajo la función f restringida al conjunto A .

1.6.2. La integral y sus propiedades

Definición 1.6.3. La **integral de Lebesgue** de una función real medible positiva $f \in \mathcal{L}^+(\Omega)$ sobre un conjunto medible $A \subset \Omega$ se define como

$$\int_A f = \mu_{n+1}[\mathcal{U}(f \cdot \chi_A)].$$

Nótese que la integral de f sobre todo su dominio Ω coincide con $\mu_{n+1}[\mathcal{U}(f)]$. Intuitivamente, vemos que este concepto extiende al de área bajo la curva de f en A para \mathbb{R}^2 , o el correspondiente volumen en el caso de \mathbb{R}^3 .

Observación 1.6.2. Una consecuencia obvia de la Def. 1.6.3 es que, dado $f \in \mathcal{L}^+(\Omega)$, se cumple que $\int_A f = \int_\Omega (f \cdot \chi_A)$, por lo que podemos reducir el estudio de la integral al dominio de definición de f . Esta propiedad de la integral se denomina **localización**. A continuación, estudiamos más propiedades importantes de la integral de Lebesgue en $\mathcal{L}^+(\Omega)$.

Propiedad 1.6.1 (Monotonía). *Dadas $f, g \in \mathcal{L}^+(\Omega)$ tal que $f(x) \leq g(x) \forall x \in \Omega$, se tiene que*

$$\int_\Omega f \leq \int_\Omega g.$$

Demostración. Evidentemente, $\mathcal{U}(f) \subset \mathcal{U}(g)$, por lo cual

$$\int_\Omega f = \mu_{n+1}[\mathcal{U}(f)] \leq \mu_{n+1}[\mathcal{U}(g)] = \int_\Omega g.$$

□

Propiedad 1.6.2 (Homogeneidad). *Dada $f \in \mathcal{L}^+(\Omega)$ y $\rho \in \mathbb{R}_0^+$,*

$$\int_\Omega \rho \cdot f = \rho \cdot \int_\Omega f.$$

Demostración. Tomando la aplicación $\phi_\rho^{(n+1)}$ como en (1.12), es claro que $\mathcal{U}(\rho \cdot f) = \phi_\rho^{(n+1)}(\mathcal{U}(f))$. Entonces, $\int_\Omega \rho \cdot f = \mu_{n+1}[\mathcal{U}(\rho \cdot f)] = \mu_{n+1}[\phi_\rho^{(n+1)}(\mathcal{U}(f))] = \rho \cdot \mu_{n+1}[\mathcal{U}(f)] = \rho \cdot \int_\Omega f$. □

Propiedad 1.6.3 (Aditividad). *Sea $f \in \mathcal{L}^+(\Omega)$ y sea una familia $\{A_j\} \subset \mathcal{M}_n$ tal que $\Omega = \bigsqcup_{j=1}^\infty A_j$. Entonces,*

$$\int_\Omega f = \sum_{j=1}^\infty \int_\Omega f \cdot \chi_{A_j}.$$

Demostración. Claramente, $\mathcal{U}(f) = \bigsqcup_{j=1}^\infty \mathcal{U}(f \cdot \chi_{A_j})$, por lo que

$$\int_\Omega f = \mu_{n+1}[\mathcal{U}(f)] = \mu_{n+1}\left[\bigsqcup_{j=1}^\infty \mathcal{U}(f \cdot \chi_{A_j})\right] = \sum_{j=1}^\infty \mu_{n+1}[\mathcal{U}(f \cdot \chi_{A_j})] = \sum_{j=1}^\infty \int_\Omega f \cdot \chi_{A_j}.$$

□

Propiedad 1.6.4 (Integral de la función característica). *Dado $A \subset \mathbb{R}^n$ medible,*

$$\int_{\mathbb{R}^n} \chi_A = \mu_n(A).$$

Demostración. En este caso, por la Propiedad 1.4.5, $\mathcal{U}(\chi_A) = A \times]0, 1[$ es medible y

$$\mu_{n+1}[\mathcal{U}(\chi_A)] = \mu_{n+1}(A \times]0, 1[) = \mu_n(A) \cdot \mu_1(]0, 1[) = \mu_n(A).$$

□

Corolario 1.6.4.1. *Sea un intervalo acotado $\tilde{I} \subset \mathcal{I}_{n+1}$, entonces*

$$\mu_{n+1}(\tilde{I}) = \int_{\mathbb{R}^n} \delta \cdot \chi_I$$

donde $I = \prod_{i=1}^n \pi_i(\tilde{I})$ es el intervalo acotado que forman las primeras n proyecciones de \tilde{I} , y $\delta = \mu_1(\pi_{n+1}(\tilde{I}))$ es la longitud de su $(n+1)$ -ésima proyección.

Demostración. Se sigue de la Propiedad 1.6.2 y la Propiedad 1.6.3, notando que la longitud δ ha de ser no negativa y el intervalo acotado I es evidentemente medible. □.

Propiedad 1.6.5 (Integrales nulas). *Sea $f \in \mathcal{L}^+(\Omega)$. Entonces,*

$$\mu_n[f^{-1}(]0, +\infty[)] = 0 \iff \int_{\Omega} f = 0.$$

Demostración. Notemos $A = f^{-1}(]0, +\infty[)$. Vemos que $A \in \mathcal{M}_n$ como imagen inversa de un abierto. Si $\mu_n(A) = 0$, entonces $\mathcal{U}(f) \subset A \times \mathbb{R}^+$, lo cual nos dice que $\mu_{n+1}[\mathcal{U}(f)] = \int_{\Omega} f = 0$ por la Proposición 1.2.4. Para la otra implicación, tomamos el conjunto $A_k = f^{-1}([\frac{1}{k}, +\infty[)$ para cada $k \in \mathbb{N}$, también medible en \mathbb{R}^n por la Observación 1.5.1. Este verifica claramente que $A = \bigcup_{k=1}^{\infty} A_k$. Dado $k \in \mathbb{N}$, por la construcción de A_k , vemos que $A_k \times]0, \frac{1}{k}[\subset \mathcal{U}(f)$, es decir,

$$\frac{\mu_n(A_k)}{k} = \mu_{n+1}(A_k \times]0, \frac{1}{k}[) \leq \mu_{n+1}[\mathcal{U}(f)] = \int_{\Omega} f = 0.$$

En consecuencia, $\mu_n(A_k) = 0 \forall k \in \mathbb{N}$, lo que nos confirma que $\lim_{k \rightarrow \infty} \mu_n(A_k) = \mu_n(A) = 0$, por la continuidad creciente de μ_n . □

Observación 1.6.3. La Propiedad 1.6.5 nos indica que hay funciones medibles positivas que no son idénticamente nulas y aún así tienen integral cero. Para ello, basta con que se anulen excepto en un conjunto de medida nula; o en particular, que Ω sea un conjunto de medida nula.

Propiedad 1.6.6 (Suma). *Sean $f, g \in \mathcal{L}^+(\Omega)$, entonces*

$$\int_{\Omega} (f + g) = \int_{\Omega} f + \int_{\Omega} g.$$

Demostración. En caso de que $\int_{\Omega} f$ o $\int_{\Omega} g$ tengan valor infinito, la igualdad es trivial pues $\mathcal{U}(f) \subset \mathcal{U}(f + g)$ y $\mathcal{U}(g) \subset \mathcal{U}(f + g)$. Para el caso finito, notamos que podemos expresar la unión disjunta de medibles

$$\mathcal{U}(f + g) = \mathcal{U}(f) \sqcup \mathcal{G}(f) \sqcup \{(x, y) \in \Omega \times \mathbb{R} : f(x) < y < g(x) + f(x)\} \quad (1.16)$$

donde el último conjunto lo es por poder expresarse como diferencia de subgráficas medibles. Como $\mu_{n+1}[\mathcal{G}(f)] = 0$, bastará con ver que el conjunto mencionado tiene la misma medida

μ_{n+1} que $\mathcal{U}(g)$, en cuyo caso

$$\int_{\Omega} (f + g) = \mu_{n+1}[\mathcal{U}(f + g)] = \mu_{n+1}[\mathcal{U}(f)] + \mu_{n+1}[\mathcal{U}(g)] = \int_{\Omega} f + \int_{\Omega} g. \quad (1.17)$$

Para ello, definimos en primer lugar la aplicación

$$\begin{aligned} \mathcal{T}_f : \mathbb{R}^n \times \mathbb{R} &\rightarrow \mathbb{R}^n \times \mathbb{R} \\ (x, y) &\mapsto \begin{cases} (x, y + f(x)) & \text{si } x \in \Omega \\ (x, y) & \text{si } x \notin \Omega. \end{cases} \end{aligned}$$

Esta aplicación es biyectiva con $\mathcal{T}_f^{-1} = \mathcal{T}_{-f}$. Definimos también $f_0 : \mathbb{R}^n \rightarrow \mathbb{R}$ con $f_0(x) = f(x) \forall x \in \Omega$ y $f_0(x) = 0 \forall x \notin \Omega$, la cual es medible positiva pues $f \in \mathcal{L}^+(\Omega)$ y $\Omega^c \in \mathcal{M}_n$. Claramente, $\mathcal{T}_f = \mathcal{T}_{f_0}$. Dada una función $h \in \mathcal{L}^+$, la aplicación \mathcal{T}_f verifica que

$$\mathcal{T}_f(\mathcal{U}(h)) = \{(x, y) \in \mathbb{R}^n \times \mathbb{R} : f_0(x) < y < h(x) + f_0(x)\}.$$

Sea ahora un intervalo acotado $\tilde{I} \in \mathcal{I}_{n+1}$. Es posible tomar $I \in \mathcal{I}_n$ y $\delta \in \mathbb{R}_0^+$ con $\mu_{n+1}(\tilde{I}) = \mu_{n+1}[\mathcal{U}(\delta \cdot \chi_I)]$, en las condiciones del [Corolario 1.6.4.1](#). Tenemos la siguiente igualdad:

$$\begin{aligned} \mathcal{U}((f_0 + \delta) \cdot \chi_I) &= \mathcal{U}(f_0 \cdot \chi_I) \sqcup \mathcal{G}(f_0 \cdot \chi_I) \sqcup \mathcal{T}_f(\mathcal{U}(\delta \cdot \chi_I)) = \\ &= \mathcal{U}(\delta \cdot \chi_I) \sqcup \mathcal{G}(\delta \cdot \chi_I) \sqcup \mathcal{T}_\delta(\mathcal{U}(f_0 \cdot \chi_I)) = \mathcal{U}((f_0 + \delta) \cdot \chi_I). \end{aligned}$$

Todos estos conjuntos son medibles, teniendo en cuenta que $f_0 \cdot \chi_I, \delta \cdot \chi_I \in \mathcal{L}^+(\Omega)$ y lo visto en [\(1.16\)](#). Nótese que la aplicación \mathcal{T}_δ es equivalente a una traslación por el punto $(0, 0, \dots, 0, \delta) \in \mathbb{R}^{n+1}$, por lo que $\mu_{n+1}[\mathcal{U}(f_0 \cdot \chi_I)] = \mu_{n+1}[\mathcal{T}_\delta(\mathcal{U}(f_0 \cdot \chi_I))]$ por la invariancia a traslaciones de μ_{n+1} . Aplicando la σ -aditividad de μ_{n+1} , el hecho de que las gráficas tienen medida nula y cancelando términos, lo cual se puede hacer ya que $\mu_{n+1}[\mathcal{U}(f_0 \cdot \chi_I)] = \int_{\Omega} f_0 \cdot \chi_I \leq \int_{\Omega} f < \infty$, nos queda

$$\mu_{n+1}[\mathcal{T}_f(\mathcal{U}(\delta \cdot \chi_I))] = \mu_{n+1}[\mathcal{U}(\delta \cdot \chi_I)] = \mu_{n+1}(\tilde{I}). \quad (1.18)$$

Suponiendo que $\alpha \in \mathbb{R}$ es el extremo izquierdo de la $(n+1)$ -ésima proyección de \tilde{I} , es fácil ver que $\mathcal{U}(\delta \cdot \chi_I) = \tilde{I} \setminus [\mathcal{G}(\alpha) \cup \mathcal{G}(\alpha + \delta)]$, y por la biyectividad de \mathcal{T}_f ,

$$\mathcal{T}_f(\mathcal{U}(\delta \cdot \chi_I)) = \mathcal{T}_f(\tilde{I}) \setminus \mathcal{T}_f(\mathcal{G}(\alpha) \cup \mathcal{G}(\alpha + \delta)) = \mathcal{T}_f(\tilde{I}) \setminus (\mathcal{G}(f_0 + \alpha) \cup \mathcal{G}(f_0 + \alpha + \delta)).$$

Entonces, $\mu_{n+1}[\mathcal{T}_f(\mathcal{U}(\delta \cdot \chi_I))] = \mu_{n+1}[\mathcal{T}_f(\tilde{I})]$, pues las gráficas anteriores tienen medida nula. Llegamos a que $\mu_{n+1}(\tilde{I}) = \mu_{n+1}[\mathcal{T}_f(\tilde{I})]$ por [\(1.18\)](#). Como $\tilde{I} \in \mathcal{I}_{n+1}$ era arbitrario, tomando $A \in \mathcal{M}_{n+1}$, podemos escoger un recubrimiento de $\{\tilde{I}_j\} \subset \mathcal{I}_{n+1}$ de A , de forma que

$$\mu_{n+1}^*[\mathcal{T}_f(A)] \leq \sum_{j=1}^{\infty} \mu_{n+1}[\mathcal{T}_f(\tilde{I}_j)] \leq \sum_{j=1}^{\infty} \mu_{n+1}(\tilde{I}_j),$$

donde la primera desigualdad se da por ser $\{\mathcal{T}_f(\tilde{I}_j)\}$ un recubrimiento de $\mathcal{T}_f(A)$. La arbitrariedad $\{\tilde{I}_j\}$ nos indica que $\mu_{n+1}^*[\mathcal{T}_f(A)] \leq \mu_{n+1}(A)$.

Tomando la aplicación de [\(1.12\)](#) con $j = n+1$ y $\rho = -1$, vemos que la transformación \mathcal{T}_{-f}

se puede expresar como $\phi_{-1}^{(n+1)} \circ \mathcal{T}_f \circ \phi_{-1}^{(n+1)}$, y usando la desigualdad recién probada,

$$\begin{aligned}\mu_{n+1}^*[\mathcal{T}_{-f}(A)] &= \mu_{n+1}^*\left[\left(\phi_{-1}^{(n+1)} \circ \mathcal{T}_f \circ \phi_{-1}^{(n+1)}\right)(A)\right] = \\ &= \mu_{n+1}^*\left[\mathcal{T}_f(\phi_{-1}^{(n+1)}(A))\right] \leq \mu_{n+1}(\phi_{-1}^{(n+1)}(A)) = \mu_{n+1}(A)\end{aligned}$$

para cada $A \in \mathcal{M}$. En particular, tomando $A = \mathcal{U}(g)$, sabemos que $\mathcal{T}_f(\mathcal{U}(g)) \in \mathcal{M}_{n+1}$ y

$$\mu_{n+1}[\mathcal{U}(g)] = \mu_{n+1}[\mathcal{T}_{-f} \circ \mathcal{T}_f(\mathcal{U}(g))] \leq \mu_{n+1}[\mathcal{T}_f(\mathcal{U}(g))] \leq \mu_{n+1}[\mathcal{U}(g)],$$

por lo que tenemos la igualdad buscada y se cumple (1.17). \square

Corolario 1.6.6.1. Si $f, g \in \mathcal{L}^+(\Omega)$ con $g(x) \leq f(x) \forall x \in \Omega$, entonces

$$\int_{\Omega}(f-g) = \int_{\Omega}f - \int_{\Omega}g.$$

Demostración. En estas condiciones, $f-g \in \mathcal{L}^+(\Omega)$ y

$$\int_{\Omega}((f-g)+g) = \int_{\Omega}(f-g) + \int_{\Omega}g \iff \int_{\Omega}(f-g) = \int_{\Omega}f - \int_{\Omega}g.$$

\square

Propiedad 1.6.7 (Convergencia monótona). Sea $\{f_j\} \subset \mathcal{L}^+(\Omega)$ una sucesión creciente de funciones medibles positivas y sea $f : \Omega \rightarrow \mathbb{R}$ su límite puntual. Entonces,

$$\int_{\Omega}f = \lim_{j \rightarrow \infty} \int_{\Omega}f_j.$$

Demostración. Como $\{f_j\}$ converge puntualmente f en Ω y es creciente, $f \in \mathcal{L}^+(\Omega)$ como consecuencia de la Proposición 1.5.2. Además, $\mathcal{U}(f_j) \subset \mathcal{U}(f_{j+1})$ para cada $j \in \mathbb{N}$, con $\mathcal{U}(f) = \bigcup_{j=1}^{\infty} \mathcal{U}(f_j)$. Por la continuidad creciente de μ_{n+1} ,

$$\mu_{n+1}[\mathcal{U}(f)] = \lim_{j \rightarrow \infty} \mu_{n+1}[\mathcal{U}(f_j)].$$

Esta igualdad equivale a la enunciada. \square .

Corolario 1.6.7.1 (Integral de la suma de una serie). Sea $\{f_j\} \subset \mathcal{L}^+(\Omega)$.

$$\int_{\Omega} \sum_{j=1}^{\infty} f_j = \sum_{j=1}^{\infty} \int_{\Omega} f_j.$$

Demostración. Para $j \in \mathbb{N}$, $S_j = \sum_{k=1}^j f_k \in \mathcal{L}^+(\Omega)$, como suma de funciones medibles positivas. Además, $S_j(x) \leq S_{j+1}(x) \forall x \in \Omega$, ya que $S_{j+1} = S_j + f_{j+1}$. Por tanto, tomando $S_{\infty} : \Omega \rightarrow \mathbb{R}$ con $S_{\infty}(x) = \lim_{j \rightarrow \infty} S_j(x) = \sum_{k=1}^{\infty} f_k(x) \forall x \in \Omega$, estamos en condiciones de la Propiedad 1.6.7. Usando la Propiedad 1.6.6, podemos desarrollar la suma de integrales para cada $j \in \mathbb{N}$, de forma que en el límite, se da la igualdad del enunciado. \square

1.6.3. Integrabilidad: el espacio $\mathcal{L}^1(\Omega)$

Definición 1.6.4. Sea $f \in \mathcal{L}(\Omega)$ una función medible. Si alguna de las integrales $\int_{\Omega} f_+$ o $\int_{\Omega} f_-$ es finita, definimos la **integral de Lebesgue de f** como la cantidad

$$\int_{\Omega} f = \int_{\Omega} f_+ - \int_{\Omega} f_-.$$

Como cualquier función $f \in \mathcal{L}(\Omega)$ puede descomponerse como $f = f_+ - f_-$, donde $f_+, f_- \in \mathcal{L}^+(\Omega)$, podemos reducir el estudio de la integral de una función medible al de las integrales de dos funciones medibles positivas.

Definición 1.6.5. Decimos que $f \in \mathcal{L}(\Omega)$ es **integrable de Lebesgue** si tanto $\int_{\Omega} f_+$ como $\int_{\Omega} f_-$ son finitas. Equivalentemente, dado que $|f| = f_+ + f_-$, f es integrable de Lebesgue si $\int_{\Omega} |f| < \infty$.

Denotamos por $\mathcal{L}^1(\Omega)$ al conjunto de las funciones integrables de Lebesgue definidas en Ω . Cuando $\Omega = \mathbb{R}^n$, dicho conjunto se notará por \mathcal{L}^1 . Se tiene que $\mathcal{L}^1(\Omega) \subset \mathcal{L}(\Omega)$.

Proposición 1.6.3. $\mathcal{L}^1(\Omega)$ es un subespacio vectorial de $\mathcal{L}(\Omega)$.

Demostración. En efecto, sean $f, g \in \mathcal{L}^1(\Omega)$, entonces $\int_{\Omega} |f| < \infty$ y $\int_{\Omega} |g| < \infty$. Por la **Propiedad 1.6.6** y la desigualdad triangular,

$$\int_{\Omega} |f + g| \leq \int_{\Omega} (|f| + |g|) = \int_{\Omega} |f| + \int_{\Omega} |g| < \infty,$$

luego $f + g \in \mathcal{L}^1(\Omega)$. Así mismo, dado $\rho \in \mathbb{R}$, por la **Propiedad 1.6.2**,

$$\int_{\Omega} |\rho \cdot f| = \int_{\Omega} (|\rho| \cdot |f|) = |\rho| \cdot \int_{\Omega} |f| < \infty,$$

luego $\rho \cdot f \in \mathcal{L}^1(\Omega)$. Hemos probado que $\mathcal{L}^1(\Omega)$ es cerrado para la suma y el producto por escalares, luego es un subespacio vectorial de $\mathcal{L}(\Omega)$. \square

1.7. Estimación mediante funciones simples

Definición 1.7.1. Una **función simple** es una función $S : \Omega \rightarrow \mathbb{R}$ que se puede escribir como una combinación lineal finita de funciones características de subconjuntos medibles de Ω . Es decir, una función simple S tiene la forma

$$S(x) = \sum_{j=1}^m \alpha_j \cdot \chi_{A_j}(x),$$

donde $m \in \mathbb{N}$, $\{A_j\}_{j=1}^m \subset \mathcal{M} \cap \mathcal{P}(\Omega)$ y $\alpha_j \in \mathbb{R}$ para cada $j \in \Gamma_m$.

A continuación, introduciremos algunos resultados que demuestran la existencia de funciones simples capaces de aproximar funciones en $\mathcal{L}^1(\Omega)$ usando una determinada noción de distancia.

1.7.1. Distancia en $\mathcal{L}^1(\Omega)$

Definición 1.7.2. Sea $f \in \mathcal{L}^1(\Omega)$, definimos

$$\|f\|_{\mathcal{L}^1(\Omega)} = \int_{\Omega} |f| < \infty.$$

La función $\|\cdot\|_{\mathcal{L}^1(\Omega)} : \mathcal{L}^1(\Omega) \rightarrow \mathbb{R}_0^+$ asociada define una **seminorma** en $\mathcal{L}^1(\Omega)$, al verificar las siguientes tres propiedades:

- **No negatividad:** $\|f\|_{\mathcal{L}^1(\Omega)} \geq 0$, ya que $|f| \in \mathcal{L}^+(\Omega)$, luego su integral es no negativa.
- **Homogeneidad:** $\|\rho \cdot f\|_{\mathcal{L}^1(\Omega)} = \int_{\Omega} |\rho| \cdot |f| = |\rho| \cdot \int_{\Omega} |f| = |\rho| \cdot \|f\|_{\mathcal{L}^1(\Omega)} \quad \forall f \in \mathcal{L}^1(\Omega), \rho \in \mathbb{R}$.
- **Desigualdad triangular:** $\|f + g\|_{\mathcal{L}^1(\Omega)} \leq \|f\|_{\mathcal{L}^1(\Omega)} + \|g\|_{\mathcal{L}^1(\Omega)} \quad \forall f, g \in \mathcal{L}^1(\Omega)$ usando la desigualdad triangular para el valor absoluto, el crecimiento y la aditividad de la integral.

Cuando $\Omega = \mathbb{R}^n$, la denotaremos simplemente por $\|\cdot\|_{\mathcal{L}^1}$.

Observación 1.7.1. Dada $f \in \mathcal{L}^1(\Omega)$ con $\|f\|_1 = 0$, la función puede ser no nula en un conjunto de puntos de medida nula, como vimos en [Propiedad 1.6.5](#). Decimos entonces que f es **casi idénticamente nula**, pero puede no ser exactamente la función constantemente igual a cero. Para salvar esta problemática y construir un espacio normado al uso, se trabaja con el espacio cociente $\mathcal{L}^1(\Omega)/\mathcal{N}$, donde

$$\mathcal{N} = \{f \in \mathcal{L}^1(\Omega) : f \text{ es casi idénticamente nula}\}.$$

Ajustando convenientemente la definición de la seminorma a las clases de equivalencia, es posible dotar de una norma a dicho espacio cociente. En nuestro contexto, no será realmente necesario realizar esta distinción.

Definición 1.7.3. La **semidistancia** en $\mathcal{L}^1(\Omega)$ asociada a $\|\cdot\|_{\mathcal{L}^1(\Omega)}$ viene dada por

$$d_1(f, g) = \|f - g\|_{\mathcal{L}^1(\Omega)} \quad \forall f, g \in \mathcal{L}^1(\Omega).$$

Dadas f y g medibles positivas e integrables en Ω , esta semidistancia se interpreta geométricamente como la medida de la diferencia simétrica entre las subgráficas de dos funciones. En concreto,

$$\begin{aligned} \|f - g\|_{\mathcal{L}^1(\Omega)} &= \int_{\Omega} |f - g| = \int_{\Omega} \max(f, g) - \int_{\Omega} \min(f, g) = \\ &= \mu_{n+1}[\mathcal{U}(f) \cup \mathcal{U}(g)] - \mu_{n+1}[\mathcal{U}(f) \cap \mathcal{U}(g)] = \mu_{n+1}[\mathcal{U}(f) \Delta \mathcal{U}(g)]. \end{aligned}$$

Por la [Observación 1.7.1](#), sabemos que si la semidistancia d_1 entre dos funciones integrables es cero, no podemos asumir que estas sean iguales: pueden no serlo en un conjunto de puntos de medida nula. Aún así, por comodidad, nos vamos a referir generalmente a este concepto con el de distancia en $\mathcal{L}^1(\Omega)$.

1.7.2. Funciones integrables

En primer lugar, observaremos cómo aproximar funciones en $\mathcal{L}^1(\Omega)$ mediante funciones simples, con respecto de la semidistancia definida en dicho espacio. Estas funciones simples serán combinaciones lineales de funciones características de cubos en \mathbb{R}^n .

Teorema 1.1. Sea $f : \Omega \rightarrow \mathbb{R}$ una función medible positiva e integrable, y sea $\epsilon > 0$.

Entonces, existe una familia finita de cubos $\{C_j\}_{j=1}^m \subset \mathbb{R}^n$, $m \in \mathbb{N}$, y escalares $\delta_1, \delta_2, \dots, \delta_m \in \mathbb{R}_0^+$ de forma que la función $S_m := \sum_{j=1}^m \delta_j \cdot \chi_{C_j} \in \mathcal{L}^1(\Omega)$ verifica que $\|f - S_m\|_{\mathcal{L}^1(\Omega)} < \epsilon$.

Demostración. Usando la regularidad exterior de μ_{n+1} , tomamos un abierto E de \mathbb{R}^{n+1} y tal que $\mu_{n+1}(E) < \mu_{n+1}[\mathcal{U}(f)] + \frac{\epsilon}{2} < \infty$, ya que $\mu_{n+1}[\mathcal{U}(f)] = \int_{\Omega} f < \infty$. El abierto E se puede expresar como unión disjunta de una familia $\{\tilde{C}_j\}$ de cubos diádicos de \mathbb{R}^{n+1} , por lo que $\sum_{j=1}^{\infty} \mu_{n+1}(\tilde{C}_j) = \mu_{n+1}(E) < \infty$. Entonces, la serie converge y $\exists m \in \mathbb{N}$ tal que $\sum_{j=m+1}^{\infty} \mu_{n+1}(\tilde{C}_j) < \frac{\epsilon}{2}$.

Para cada $j \in \mathbb{N}$, tomamos el cubo $C_j = \prod_{i=1}^n \pi_i(\tilde{C}_j) \in \mathcal{I}_n$, y sea $\delta_j \in \mathbb{R}_0^+$ la longitud de cada lado de este. Usando el [Corolario 1.6.4.1](#), tenemos que

$$\mu_{n+1}(\tilde{C}_j) = \int_{\mathbb{R}^n} \delta_j \cdot \chi_{C_j}.$$

Para el $m \in \mathbb{N}$ anterior, definimos $S_m := \sum_{j=1}^m \delta_j \cdot \chi_{C_j} : \Omega \rightarrow \mathbb{R}$ y sea $S_{\infty} = \lim_{m \rightarrow \infty} S_m$. $S_m \in \mathcal{L}^+(\Omega)$ como suma de funciones medibles positivas, y S_{∞} también como límite puntual creciente de una sucesión de funciones medibles positivas. Por el [Corolario 1.6.7.1](#),

$$\int_{\Omega} S_m \leq \int_{\Omega} S_{\infty} = \sum_{j=1}^{\infty} \int_{\Omega} \delta_j \cdot \chi_{C_j} \leq \sum_{j=1}^{\infty} \mu_{n+1}(\tilde{C}_j) = \mu_{n+1}(E) < \infty,$$

lo cual nos dice que $S_m, S_{\infty} \in \mathcal{L}^1(\Omega)$. Se cumple pues que

$$\|S_{\infty} - S_m\|_{\mathcal{L}^1(\Omega)} = \int_{\Omega} \sum_{j=m+1}^{\infty} \delta_j \cdot \chi_{C_j} = \sum_{j=m+1}^{\infty} \int_{\Omega} \delta_j \cdot \chi_{C_j} \leq \sum_{j=m+1}^{\infty} \mu_{n+1}(\tilde{C}_j) < \frac{\epsilon}{2}.$$

Como $\{\tilde{C}_j\}$ recubre a $\mathcal{U}(f)$, dado un $x \in \Omega$, existe un subconjunto $Q_x \subset \mathbb{N}$ tal que la subfamilia disjunta $\{\tilde{C}_j\}_{j \in Q_x}$ recubre a $\mathcal{U}(f \cdot \chi_{\{x\}})$ verificando que $\tilde{C}_j \cap \mathcal{U}(f \cdot \chi_{\{x\}})$ para cada $j \in Q_x$, lo cual implica que $x \in C_j$. Entonces, $f(x) = \mu_1([0, f(x)]) \leq \mu_1(\bigcup_{j \in Q_x} \pi_{n+1}(\tilde{C}_j)) = \sum_{j \in Q_x} \mu_1(\pi_{n+1}(\tilde{C}_j)) = \sum_{j \in Q_x} \delta_j \leq \sum_{j=1}^{\infty} \delta_j \cdot \chi_{C_j}(x) = S_{\infty}(x)$. Por tanto, $S_{\infty} - f \in \mathcal{L}^+(\Omega)$ y aplicando el [Corolario 1.6.6.1](#),

$$\|f - S_{\infty}\|_{\mathcal{L}^1(\Omega)} = \int_{\Omega} (S_{\infty} - f) = \int_{\Omega} S_{\infty} - \int_{\Omega} f \leq \mu_{n+1}(E) - \mu_{n+1}[\mathcal{U}(f)] < \frac{\epsilon}{2}.$$

Por último, usando la desigualdad triangular:

$$\|f - S_m\|_{\mathcal{L}^1(\Omega)} \leq \|f - S_{\infty}\|_{\mathcal{L}^1(\Omega)} + \|S_{\infty} - S_m\|_{\mathcal{L}^1(\Omega)} < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon.$$

□

Corolario 1.1.1. Sea $f \in \mathcal{L}^1(\Omega)$ y $\epsilon > 0$. Entonces, existe una función $S \in \mathcal{L}^1(\Omega)$ que es una combinación lineal finita de funciones características de cubos de \mathbb{R}^n tal que $\|f - S\|_{\mathcal{L}^1(\Omega)} < \epsilon$.

Demostración. Aplicando el [Teorema 1.1](#) a f_+ y f_- , que son medibles positivas e integrables, encontramos dos combinaciones lineales de funciones características de cubos cerrados: S_{m_1} con $\|f_+ - S_{m_1}\|_{\mathcal{L}^1(\Omega)} < \epsilon/2$ y S_{m_2} tal que $\|f_- - S_{m_2}\|_{\mathcal{L}^1(\Omega)} < \epsilon/2$, $m_1, m_2 \in \mathbb{N}$.

La función $S := S_{m_1} - S_{m_2}$ es también una combinación lineal de $m_1 + m_2$ funciones características de cubos cerrados, y es integrable por ser sus términos integrables y $\mathcal{L}^1(\Omega)$ un subespacio vectorial. Viendo que $f = f_+ - f_-$, se tiene que

$$\begin{aligned}\|f - S\|_{\mathcal{L}^1(\Omega)} &= \|(f_+ - S_{m_1}) - (f_- - S_{m_2})\|_{\mathcal{L}^1(\Omega)} \\ &\leq \|(f_+ - S_{m_1})\|_{\mathcal{L}^1(\Omega)} + \|(f_- - S_{m_2})\|_{\mathcal{L}^1(\Omega)} < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon.\end{aligned}$$

□

1.7.3. Funciones continuas

Denotamos por $\mathcal{C}(\Omega)$ al espacio de las funciones continuas definidas en Ω que toman valores reales. Notemos que, en general, pueden existir funciones de $\mathcal{C}(\Omega)$ que no estén en $\mathcal{L}^1(\Omega)$. Por ejemplo, una función constante no nula definida en \mathbb{R}^n tiene integral de Lebesgue infinita. Sin embargo, si K es un compacto de \mathbb{R}^n , sabemos que este es cerrado y acotado, y además cualquier función $f \in \mathcal{C}(K)$ alcanza un máximo y un mínimo en K , en virtud del *Teorema de Weierstraß*. En consecuencia, se puede escribir

$$\begin{aligned}\int_K f^+ &\leq \mu_{n+1} \left(K \times]0, \max_{x \in K} \{f(x)\}[\right) = \mu_n(K) \cdot \max_{x \in K} \{f(x)\} < \infty \\ \text{y } \int_K f^- &\leq \mu_{n+1} \left(K \times]0, \min_{x \in K} \{f(x)\}[\right) = \mu_n(K) \cdot \min_{x \in K} \{f(x)\} < \infty.\end{aligned}$$

Esto nos dice que $\mathcal{C}(K) \subset \mathcal{L}^1(K)$ para cualquier $K \subset \mathbb{R}^n$ compacto. Además, esta clase de funciones verifica una importante propiedad:

Propiedad 1.7.1 (Heine-Cantor). *Sea $f \in \mathcal{C}(K)$, con $K \subset \mathbb{R}^n$ compacto. Entonces, f es uniformemente continua en K , es decir:*

$$\forall \epsilon > 0 \exists \delta > 0 \text{ tal que } \|x - y\| < \delta \implies |f(x) - f(y)| < \epsilon \quad \forall x, y \in K, \quad (1.19)$$

donde $\|\cdot\|$ denota una norma en \mathbb{R}^n .

Demostración. Fijamos un $\epsilon > 0$ arbitrario y la norma $\|\cdot\|$ del máximo en \mathbb{R}^n . Como f es continua en K , dado un $x_0 \in K$, se puede encontrar $\delta_{x_0} > 0$ tal que si $x \in K$ cumple que $\|x - x_0\| < 2\delta_{x_0}$, entonces $|f(x) - f(x_0)| < \frac{\epsilon}{2}$. Es claro pues que

$$K \subset \bigcup_{x \in K} \left(\prod_{i=1}^n [\pi_i(x) - \delta_x, \pi_i(x) + \delta_x] \right).$$

Notamos que, al ser K compacto, $\exists m \in \mathbb{N}$ y $\{x_k\}_{k=1}^m \subset K$ tal que la familia finita de intervalos de \mathbb{R}^n dada por

$$I_k = \prod_{i=1}^n [\pi_i(x_k) - \delta_k, \pi_i(x_k) + \delta_k] \quad \text{para cada } k \in \Gamma_m,$$

donde $\delta_k := \delta_{x_k}$, es un recubrimiento de K . Supongamos $x, y \in K$ tales que $\|x - y\| < \delta = \min_{k \in \Gamma_m} \{\delta_k\}$. Evidentemente, $x \in I_k$ para cierto $k \in \Gamma_m$, luego $\|x - x_k\| < \delta_k$. De esta forma,

sabemos que $|f(x) - f(x_k)| < \frac{\epsilon}{2}$, por la continuidad de f en $x_k \in K$. Por la desigualdad triangular, $\|y - x_k\| = \|y - x\| + \|x - x_k\| < \delta + \delta_k \leq 2\delta_k$. De nuevo por continuidad, tenemos que $|f(y) - f(x_k)| < \frac{\epsilon}{2}$. Por último, aplicando de nuevo la desigualdad triangular:

$$|f(x) - f(y)| = |f(x) - f(x_k)| + |f(x_k) - f(y)| < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon$$

La arbitrariedad de ϵ y de los puntos $x, y \in K$ nos da la continuidad uniforme de f en K . \square

A partir de esta propiedad, es sencillo razonar cómo aproximar arbitrariamente una función de $\mathcal{C}(K)$ mediante funciones simples.

Proposición 1.7.1. *Sea $f \in \mathcal{C}(K)$, con $K \subset \mathbb{R}^n$ compacto, y un $\epsilon > 0$. Entonces, existe una familia $\{P_j\}_{j=1}^m \subset \mathcal{P}(K)$, con $m \in \mathbb{N}$, tal que $K = \bigcup_{j=1}^m P_j$ verificando que*

$$|f(x) - S_m(x)| < \epsilon \quad \forall x \in K, \tag{1.20}$$

donde $S_m := \sum_{j=1}^m f(c_j) \cdot \chi_{P_j} : K \rightarrow \mathbb{R}$, con $c_j \in P_j$ para cada $j \in \Gamma_m$. Además,

$$\|f - S_m\|_{\mathcal{L}^1(K)} < \mu(K) \cdot \epsilon. \tag{1.21}$$

Demostración. Como los conjuntos compactos $K \subset \mathbb{R}^n$ son trivialmente acotados, existe $N > 0$ tal que $K \subset [-N, N]^n$. Sea $k \in \mathbb{N}$ arbitrario, definimos $\alpha_l = -N + (l - 1) \cdot \frac{2N}{k}$, donde $l \in \Gamma_{k+1}$. Ahora, dados $l_1, l_2, \dots, l_n \in \Gamma_k$, definimos siguiente intervalo de \mathbb{R}^n :

$$Q_{l_1, l_2, \dots, l_n} = [\alpha_{l_1}, \alpha_{l_1+1}] \times [\alpha_{l_2}, \alpha_{l_2+1}] \times \cdots \times [\alpha_{l_n}, \alpha_{l_n+1}].$$

Sea $\{P_j\}_{j=1}^m$ la familia formada por todos los conjuntos no vacíos de la forma $Q_{l_1, l_2, \dots, l_n} \cap K$, con $l_1, l_2, \dots, l_n \in \Gamma_k$. Evidentemente, $m \leq k^n$, por lo que la familia es finita. De este modo, es claro que $K = K \cap [-N, N]^n = \bigcup_{j=1}^m P_j$. Ahora, dado un $\epsilon > 0$ fijo, por la [Propiedad 1.7.1](#), sabemos que f es uniformemente continua, luego existe un $\delta > 0$ tal que si $x, y \in K$ verifican que $\|x - y\| < \delta$, entonces $|f(x) - f(y)| < \epsilon$. Basta pues con tomar $k = \lceil \frac{2N}{\delta} \rceil$. En tal caso, si $x, y \in P_j$ para algún $j \in \Gamma_m$, entonces $\|x - y\| < \delta$ (con la norma del máximo), por lo que $|f(x) - f(y)| < \epsilon$. Entonces, vamos a seleccionar un punto $c_j \in P_j$ para cada $j \in \Gamma_m$. Usando que $\{P_j\}_{j=1}^m$ es una partición de K , concluimos que para cada $x \in K$, se verifica que $x \in P_j$ para un único $j \in \Gamma_m$, luego $\chi_{P_j}(x) \neq 0$ para un único $j \in \Gamma_m$, cumpliéndose además que $|f(x) - f(c_j)| < \epsilon$. Este razonamiento nos da la desigualdad (1.20).

Para probar (1.21), basta con argumentar que $\|f - S_m\|_{\mathcal{L}^1(K)} = \int_K |f - S_m| < \int_K \epsilon = \mu(K) \cdot \epsilon$, por el crecimiento de la integral. \square

Observación 1.7.2. Nótese que, por lo visto en (1.21), el problema de minimizar la distancia de una función continua definida en un compacto K a una función simple en $\mathcal{L}^1(K)$ se puede reducir al de minimizar la distancia entre ambas en cada punto de K .

1.7.4. Funciones lipschitzianas

De la [Subsección 1.7.3](#), extraemos la conclusión de que siempre existen funciones simples que aproximan cualquier función de $\mathcal{C}(K)$, con $K \subset \mathbb{R}^n$ compacto, hasta cualquier grado de precisión, tanto puntualmente como en términos de la distancia en $\mathcal{L}^1(K)$.

Sin embargo, el número de términos de la función simple, o lo que es lo mismo, el número de conjuntos de la partición de K que realizamos, está determinado por un factor $\delta > 0$

que depende de la elección del $\epsilon > 0$ dado y la función concreta que manejamos, como observamos en (1.19). Sería deseable establecer unas condiciones bajo las cuales podamos acotar el número de términos de la suma de forma que se garantice un error menor que un ϵ dado en la aproximación, ya sea puntual o integral, independientemente de la función seleccionada. Este marco nos lo proporcionan las llamadas funciones lipschitzianas.

Definición 1.7.4. Una función $f : \Omega \rightarrow \mathbb{R}$ se dice **lipschitziana** si se verifica que

$$|f(x) - f(y)| \leq L \cdot \|x - y\| \quad \forall x \in \Omega \quad (1.22)$$

para alguna constante $L > 0$, a la que llamamos **constante de Lipschitz**. La definición es válida usando cualquier norma $\|\cdot\|$ de \mathbb{R}^n .

Observación 1.7.3. Es sencillo comprobar que las funciones lipschitzianas con constante de Lipschitz $L > 0$ son uniformemente continuas, pues para cada $\epsilon > 0$, basta con tomar $\delta = \frac{\epsilon}{L}$, y esto verifica la condición (1.19). La continuidad, por tanto, también está garantizada.

Observación 1.7.4. Sea $A \subset \mathbb{R}^n$ acotado, y $f : A \rightarrow \mathbb{R}$ continua y derivable (excepto a lo sumo en un conjunto de puntos de medida nula \mathcal{N}). Si la derivada está acotada, es decir, existe una constante $M > 0$ tal que $|f'(x)| \leq M \forall x \in A \setminus \mathcal{N}$, entonces f es una función lipschitziana con constante de Lipschitz $L = M$.

Observación 1.7.5. Sea $A \subset \mathbb{R}^n$ acotado y medible, la imagen de una función lipschitziana $f : A \rightarrow \mathbb{R}$ de constante de Lipschitz $L > 0$ está acotada, pues sea $N > 0$ con $A \subset [-N, N]^n$, $\|x - y\| \leq 2N$ (usando la norma del máximo), luego $|f(x) - f(y)| \leq 2N \cdot L \forall x \in A$. Este hecho también nos confirma que $f \in \mathcal{L}^1(A)$.

Proposición 1.7.2. *Sea una función lipschitziana $f : A \rightarrow \mathbb{R}$ con constante de Lipschitz $L > 0$, siendo $A \subset \mathbb{R}^n$ acotado y medible, y un $\epsilon > 0$. Sea $N > 0$ tal que $A \subset [-N, N]^n$. Entonces, existe una familia $\{P_j\}_{j=1}^m \subset \mathcal{P}(A)$ tal que $A = \bigsqcup_{j=1}^m P_j$ y verificando que*

$$m \leq \left\lceil \frac{2N \cdot L}{\epsilon} \right\rceil^n \quad y \quad \mu(P_j) \leq \left(\frac{\epsilon}{L} \right)^n \quad \forall j \in \Gamma_m,$$

de forma que se cumple que

$$|f(x) - S_m(x)| < \epsilon \quad \forall x \in A,$$

donde $S_m = \sum_{j=1}^m f(c_j) \cdot \chi_{P_j} : A \rightarrow \mathbb{R}$, con $c_j \in P_j$ para cada $j \in \Gamma_m$. Además,

$$\|f - S_m\|_{\mathcal{L}^1(K)} < \mu(A) \cdot \epsilon.$$

Demostración. La demostración es análoga a la de la Proposición 1.7.1, basta con usar la Observación 1.7.5 para ver que en este caso, $\delta = \frac{\epsilon}{L}$, luego tomamos $k = \lceil \frac{2N \cdot L}{\epsilon} \rceil$, y observamos que el número de elementos de la partición es tal que $m \leq k^n = \left\lceil \frac{2N \cdot L}{\epsilon} \right\rceil^n$. Así mismo, si $x, y \in A$ pertenecen a un conjunto de la partición, serán tales que $\|x - y\| < \frac{\epsilon}{L}$ con la norma del máximo, luego la medida de cada conjunto será, a lo sumo, $(\frac{\epsilon}{L})^n$. \square

De esta forma, hemos establecido un número máximo $m \in \mathbb{N}$ de términos que necesitaremos en una función simple para que aproxime, hasta cierto grado, **cualquier función lipschitziana** definida en un conjunto acotado y medible, y con una cierta constante de Lipschitz. Ahondaremos en esta idea en el Capítulo 4, aplicándola a la construcción de redes neuronales explícitas que sean manejables en términos prácticos.

2 Fundamentos de Redes Neuronales

Una **red neuronal** es un modelo computacional que se inspira en la forma en que funcionan las redes neuronales biológicas del cerebro humano. Estas redes están compuestas por unidades básicas de procesamiento llamadas **neuronas**. Cada neurona está conectada con otras mediante enlaces que transmiten señales, que se cuantifican como números reales. Estas conexiones se caracterizan por tener un peso, que es ajustable y determina la fuerza y la importancia de la conexión.

2.1. Componentes de una red neuronal

2.1.1. Neuronas

La **neurona** es la unidad básica de procesamiento en una red neuronal. Cada neurona recibe varias entradas o **inputs**, los combina linealmente, aplica una operación sobre el resultado (la cual no tiene por qué ser lineal) y pasa su salida a otras neuronas. En esencia, se pueden observar como funciones que toman varios valores de entrada y calculan un único valor de salida u **output**.

2.1.2. Pesos y biases

Los **pesos** y los **biases** son los parámetros fundamentales que determinan el funcionamiento de las redes neuronales:

- **Pesos:** Cada conexión entre dos neuronas en la red tiene un peso asociado que ajusta la señal que se transmite entre estas neuronas. En la práctica, estos pesos actúan como multiplicadores que ponderan la relevancia de cierto input en la salida de una neurona.
- **Biases:** El bias es un término adicional que se resta a la combinación lineal de las señales escaladas mediante los pesos. A menudo, se interpreta como un umbral que filtra la señal de salida de una neurona.

2.1.3. Capas

Las neuronas en una red neuronal se organizan en capas:

- **Capa de entrada:** Recibe las señales de entrada del exterior y las transmite a las capas más profundas.
- **Capas ocultas:** Estas capas intermedias realizan la mayor parte del procesamiento a través de una combinación compleja de las entradas. La cantidad de capas ocultas y su tamaño pueden variar dependiendo del problema y la complejidad del modelo.
- **Capa de salida:** Es la última capa de la red, que devuelve el resultado del procesamiento de la red, el cual estará compuesto de uno o varios outputs, dependiendo del problema concreto.

2.1.4. Funciones de activación

La **función de activación** en una neurona define cómo se transforma la suma ponderada de la entrada en una salida que se pasa a la siguiente capa. La elección de la función de activación puede afectar significativamente al rendimiento y la capacidad de la red para representar relaciones complejas.

Definición 2.1.1. Una de las funciones de activación que aparece con mayor frecuencia en la literatura es la **sigmoide**, cuya expresión es

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad \forall x \in \mathbb{R}.$$

Algunas propiedades matemáticas de esta función son:

- σ es continua e infinitamente derivable en \mathbb{R} .
- σ toma valores en $]0, 1[$. Además,

$$\sigma(x) = \begin{cases} 0 & \text{si } x \rightarrow -\infty \\ 1 & \text{si } x \rightarrow +\infty \end{cases} \quad (2.1)$$

Las funciones que verifican esta propiedad se denominan **funciones de aplastamiento**.

- σ es estrictamente creciente, ya que su derivada σ' dada por

$$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x)) \quad \forall x \in \mathbb{R} \quad (2.2)$$

nunca se anula y siempre es positiva.

- σ crece cada vez más en $] -\infty, 0[$, y su crecimiento va disminuyendo en $]0, +\infty[$. En efecto, su segunda derivada σ'' es

$$\sigma''(x) = \sigma'(x) \cdot (1 - 2\sigma(x)) \quad \forall x \in \mathbb{R}.$$

Esta expresión se anula en $x = 0$. Cuando $x < 0$, el término $(1 - 2\sigma(x))$ es negativo; y cuando $x > 0$, es positivo. Como σ' nunca se anula, concluimos que σ es cóncava en $] -\infty, 0[$ y convexa en $]0, +\infty[$.

- La gráfica de σ es simétrica con respecto del punto $(0, \frac{1}{2})$. Esto es,

$$\sigma(-x) = \frac{1}{1 + e^x} = \frac{e^{-x}}{1 + e^{-x}} = 1 - \sigma(x).$$

Definición 2.1.2. En nuestro estudio, nos centraremos en la función de activación **ReLU** (Rectified Linear Unit), que se utiliza ampliamente en las arquitecturas modernas. Su forma es la siguiente:

$$\text{ReLU}(x) = (x)^+ = \max\{0, x\} \quad \forall x \in \mathbb{R}.$$

El único efecto de esta función es truncar a cero los valores negativos. En una neurona, se aplicará sobre la combinación lineal de las señales de entrada menos el bias. Es sencillo ver que, si la combinación lineal es negativa o su valor no supera al del bias, la neurona tendrá

cero como salida. En caso contrario, la función ReLU no modificará el valor de salida de la neurona.

Existen muchas otras funciones de activación que se utilizan en la práctica: la **tangente hiperbólica**, la función **SoftMax** o modificaciones de ReLU tales como **Leaky ReLU**. Todas ellas se examinan en profundidad en artículos como [57]. La elección de la misma para el diseño de una arquitectura depende de las características del problema que se quiera resolver.

En un modelo básico de red neuronal que no utiliza funciones de activación no lineales, la salida es simplemente una combinación lineal de las entradas, lo que limita la capacidad de representación del modelo. Por tanto, la introducción de estas funciones en el flujo de una red neuronal les permite una mayor efectividad para representar funciones variadas, aumentando así su potencia expresiva. Este hecho queda de manifiesto en estudios como [28].

2.2. Redes Neuronales Totalmente Conectadas

Definición 2.2.1. Una red neuronal **totalmente conectada**, también conocida como **red de alimentación directa** o *Feedforward Neural Network (FNN)*, se caracteriza porque todas las neuronas de una capa están conectadas a cada neurona de la capa siguiente, y no existen conexiones hacia capas anteriores.

En concreto, una **red neuronal ReLU totalmente conectada** es una red neuronal de este tipo cuyas capas ocultas están conformadas por neuronas que aplican únicamente funciones de activación ReLU. Nótese que esto garantiza que las capas intermedias sólo ofrecen valores positivos como salida.

2.2.1. Representación matemática

A continuación, establecemos un estándar para representar redes neuronales de este tipo como objetos matemáticos. Pese a que la presentación se realiza en términos de funciones de activación ReLU, esta función es intercambiable por cualquier otra.

Sea \mathcal{A} una red neuronal ReLU totalmente conectada. Notamos que dicha red tomará un número $n \in \mathbb{N}$ de inputs y devolverá un número $m \in \mathbb{N}$ de valores de salida. Entonces, podemos identificar dicha red con una función $\mathcal{A} : \mathbb{R}^n \rightarrow \mathbb{R}^m$. A menudo, se hablará de una red neuronal y la función que realiza indistintamente.

Definición 2.2.2. Como hemos visto anteriormente, la red neuronal \mathcal{A} se organiza en un número $d \in \mathbb{N}$ de capas. Llamaremos **profundidad** de la red \mathcal{A} a dicho valor d .

Notamos por $\mathcal{A}^{(k)}$ a la k -ésima capa de la red neuronal \mathcal{A} , donde $k \in \Gamma_d \cup \{0\}$. La capa $\mathcal{A}^{(0)}$ será la capa de entrada, que simplemente transmite el input de la red \mathcal{A} . Por otro lado, $\mathcal{A}^{(d)}$ será la capa de salida. El resto serán las capas intermedias u ocultas.

Definición 2.2.3. Cada capa puede tener un número distinto de neuronas. La anchura w_k de la capa $\mathcal{A}^{(k)}$ es el número de neuronas que la conforman, donde $k \in \Gamma_d \cup \{0\}$. Nótese que $w_0 = n$ y $w_d = m$. Llamaremos **anchura** w de la red \mathcal{A} al máximo de las anchuras entre sus capas, excluyendo la de entrada. Es decir, $w = \max_{k \in \Gamma_d} \{w_k\}$.

2 Fundamentos de Redes Neuronales

En este sentido, la capa $\mathcal{A}^{(k)}$ es, básicamente, una función $\mathcal{A}^{(k)} : \mathbb{R}^{w_{k-1}} \rightarrow \mathbb{R}^{w_k}$. Es decir, se generan w_k salidas, una por neurona, operando con los w_{k-1} valores de salida de la anterior capa $\mathcal{A}^{(k-1)}$. La red neuronal completa está formada por la concatenación de todas estas funciones, es decir:

$$\mathcal{A} = \mathcal{A}^{(d)} \circ \mathcal{A}^{(d-1)} \circ \dots \circ \mathcal{A}^{(1)}.$$

Además, denotamos por $\mathcal{A}_i^{(k)}$ a la i -ésima neurona de la k -ésima capa, donde $k \in \Gamma_d \cup \{0\}$ e $i \in \Gamma_{w_k}$. Para $k \in \Gamma_d$ e $i \in \Gamma_{w_k}$, la neurona $\mathcal{A}_i^{(k)}$ es, básicamente, una función $\mathcal{A}_i^{(k)} : \mathbb{R}^{w_{k-1}} \rightarrow \mathbb{R}$. De hecho, se verifica que

$$\mathcal{A}^{(k)}(x) = (\mathcal{A}_1^{(k)}(x), \mathcal{A}_2^{(k)}(x), \dots, \mathcal{A}_{w_k}^{(k)}(x)) \quad \forall x \in \mathbb{R}^{w_{k-1}}. \quad (2.3)$$

Además, dada una entrada $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$ para la red \mathcal{A} , $\mathcal{A}_1^{(0)}(x) = x_i \forall i \in \Gamma_n$, por lo que $\mathcal{A}^{(0)}(x) = (x_1, x_2, \dots, x_n)$.

Por último, veremos el aspecto de las funciones que realiza cada neurona. Para ello, primero establecemos los parámetros, es decir, los pesos y los biases. Dado $k \in \Gamma_d$, la capa $\mathcal{A}^{(k)}$ tendrá asociada una matriz de pesos de dimensiones $w_k \times w_{k-1}$, la cual notaremos por $\mathcal{W}^{(k)}$. Esta matriz almacenará los multiplicadores de todas las conexiones entre neuronas de la capa $\mathcal{A}^{(k-1)}$ y la capa $\mathcal{A}^{(k)}$. En concreto, nos referiremos al peso que multiplica al valor de salida de la neurona $\mathcal{A}_j^{(k-1)}$ en la neurona $\mathcal{A}_i^{(k)}$ por $\mathcal{W}_{i,j}^{(k)}$ para cada $i \in \Gamma_{w_k}$ y $j \in \Gamma_{w_{k-1}}$.

Así mismo, establecemos el vector de biases $\mathcal{B}^{(k)}$ en dicha capa, que tendrá dimensión w_k . Notamos por $\mathcal{B}_i^{(k)}$ a la i -ésima componente de este vector, con $i \in \Gamma_{w_k}$, que se utiliza en la neurona $\mathcal{A}_i^{(k)}$.

Veamos, ahora sí, la descripción de la función que realiza la neurona $\mathcal{A}_i^{(k)}$ para cada $k \in \Gamma_d$ e $i \in \Gamma_{w_k}$, que se define recursivamente para un $x \in \mathbb{R}^n$ que toma la red como entrada por:

$$\mathcal{A}_i^{(k)}(x) = \left(\left(\sum_{j=1}^{w_{k-1}} \mathcal{W}_{i,j}^{(k)} \cdot \mathcal{A}_j^{(k-1)}(x) \right) - \mathcal{B}_i^{(k)} \right)^+.$$

De manera más compacta, podemos describir la función realizada por la capa completa $\mathcal{A}^{(k)}$, para cierto $k \in \Gamma_d$, por

$$\mathcal{A}^{(k)}(x) = \left(\mathcal{W}^{(k)} \cdot \mathcal{A}^{(k-1)}(x) - \mathcal{B}^{(k)} \right)^+ \quad \forall x \in \mathbb{R}^n,$$

donde $+$ en este caso denota la aplicación de la función de activación ReLU a un punto de \mathbb{R}^{w_k} componente a componente. Como vemos, esta expresión coincide con lo expuesto en (2.3) y nos permite calcular el output de la red \mathcal{A} en d pasos.

Para que esta representación tenga sentido, todos los pesos han de estar definidos. Si la salida de una neurona no tiene repercusión en el cálculo de la salida de otra, simplemente se define el peso correspondiente como nulo, por defecto.

2.2.2. Representación gráfica

Para representar la red neuronal \mathcal{A} descrita en el Subsección 2.2.1, utilizamos un diagrama como el de la Figura 2.1. Se siguen las siguientes directrices:

- Los valores de entrada y salida de la red se señalizan con un rombo negro.

- Las neuronas de cada capa se representarán utilizando círculos de fondo blanco. Dentro de cada uno de ellos se indica el nombre de la neurona, o bien la representación concreta de la función que realiza.
- Se incluyen flechas entre las capas contiguas que unen todas las neuronas de la primera con todas las de la segunda. En ellas se puede indicar el nombre del peso correspondiente o alternativamente, su valor explícito. Si el valor de algún peso fuese cero, podría suprimirse la flecha correspondiente.
- En cada capa, se incluye un círculo de color grisáceo que representa el efecto de los biases. Desde este salen flechas a cada una de las neuronas de la capa, donde se pueden indicar los valores de bias correspondientes.
- En el eje horizontal se indica la capa a la que pertenece cada neurona; y en el eje vertical, el número de neurona dentro de cada capa.

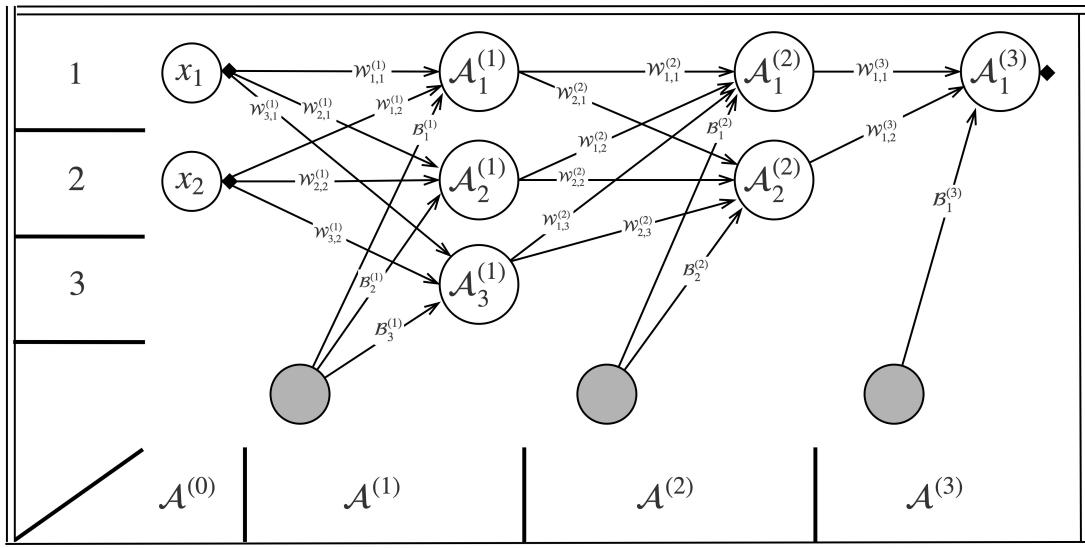


Figura 2.1: Esquema representativo de la arquitectura de una red neuronal \mathcal{A} con anchura $w = 3$ y profundidad $d = 3$.

2.3. Aprendizaje automático en redes neuronales

En el contexto de las redes neuronales, el **aprendizaje automático** se refiere al uso de algoritmos para ajustar los pesos y los biases de una arquitectura neuronal específica, de modo que esta pueda definir un modelo capaz de relacionar eficientemente unos datos que se proveen, aprendiendo patrones complejos entre ellos. La clave del éxito de las redes neuronales radica en la capacidad de estos algoritmos para adaptar los parámetros correctamente.

Concretamente, las redes neuronales se encuadran en el ámbito del **aprendizaje supervisado**, pues debemos alimentar al algoritmo de optimización concreto con un conjunto de **datos etiquetados** (pares que relacionan una entrada con una salida) suficientemente representativo, de forma que se establezca un modelo robusto que los enlace. El objetivo es que, tras un

número concreto de pasos del algoritmo, la red neuronal sea capaz de realizar predicciones coherentes a partir de nuevas entradas que no estén etiquetadas.

En efecto, buscamos que la función matemática definida por la red, que compone combinaciones lineales con funciones de activación no lineales a través de las capas (de la manera que se describe en Subsección 2.2.1), mapee las entradas en salidas deseadas. Por tanto, si seleccionamos un problema de la vida real para ser resuelto por una red, hemos de asumir que este sigue algún tipo de **distribución subyacente**. Por ejemplo, en un problema de clasificación de imágenes, esperamos que las características de las imágenes (píxeles, colores, formas...) se distribuyan de manera que puedan ser "capturadas" por la red neuronal para obtener predicciones generales satisfactorias sobre algún aspecto concreto de las mismas.

Por tanto, para un problema en el que queremos predecir $m \in \mathbb{N}$ valores a partir de $n \in \mathbb{N}$ entradas, podemos considerar que el conjunto de datos etiquetados tiene la forma $\{(x_i, f(x_i))\}_{i=1}^k \subset \mathbb{R}^n \times \mathbb{R}^m$, donde $k \in \mathbb{N}$ es el número de datos que tenemos, $\{x_i\}_{i=1}^k \subset \mathbb{R}^n$ es el conjunto de entradas que se etiquetan y $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ es la función que siguen las instancias del problema, esto es, el mapa que queremos modelizar. Nótese que consideramos que las entradas y salidas vienen dadas en términos de números reales, por lo que se necesita establecer un paralelismo entre objetos del mundo real y estos, que se plantea de distinta manera según el caso.

2.3.1. Funciones de pérdida

Los algoritmos de ajuste de parámetros en redes neuronales funcionan de forma que intentan, hasta cierto punto, minimizar el error entre las entradas y las salidas para los datos etiquetados, para lo cual hemos de elegir una manera de medir este error mediante una **función de pérdida**. Para ello, distinguimos dos tipos principales de problemas según el tipo de salida que ha de tener la red neuronal asociada:

- **Regresión:** Son problemas donde las salidas son **valores continuos**. Un ejemplo típico es la predicción de precios de viviendas basada en características como el tamaño, la ubicación y el número de habitaciones.

La función de pérdida comúnmente utilizada en estos casos es el **Error Cuadrático Medio** (MSE, por sus siglas en inglés), que se define como la media de los cuadrados de las diferencias entre los valores predichos y los valores reales:

$$\text{MSE} = \frac{1}{k} \sum_{i=1}^k (f(x_i) - \mathcal{A}(x_i))^2 \quad (2.4)$$

donde $\{(x_i, f(x_i))\}_{i=1}^k$ son los datos etiquetados y \mathcal{A} la función realizada por la red neuronal.

Esta función de pérdida penaliza los grandes errores de predicción de manera más significativa que los pequeños errores. Una desventaja es que puede resultar bastante sensible a los datos atípicos, y por tanto el modelo tendrá mayores dificultades para lidiar con ellos.

- **Clasificación:** Problemas donde la salida es una **etiqueta discreta** elegida de entre un conjunto finito de ellas. Un ejemplo clásico es la clasificación de imágenes de dígitos escritos a mano, donde cada imagen debe ser categorizada como uno de los dígitos del

o al 9, como se expone en [16]. Si tan solo existen dos etiquetas posibles, entonces el problema es de **clasificación binaria**.

Para la implementación de redes neuronales acordes a este tipo de problemas, es común considerar una capa de salida con m neuronas, donde m es el número de clases posibles. Cada neurona en la capa de salida produce una puntuación que se transforma en una probabilidad mediante la función **SoftMax**. La función SoftMax asegura que la suma de las probabilidades en las m salidas sea uno. Esto facilita la interpretación de las salidas del modelo como probabilidades de pertenencia a cada clase.

Las funciones de pérdida más comunes en este caso son la **entropía cruzada** (cross-entropy) y la **pérdida logarítmica** (log loss). Estas funciones de pérdida miden la diferencia entre las distribuciones de probabilidad predichas por la red y las distribuciones de probabilidad reales, penalizando las predicciones incorrectas más severamente.

2.3.2. Metodología de entrenamiento

El **entrenamiento** de una red neuronal corresponde al proceso de minimización de la función de pérdida para un subconjunto de los datos etiquetados de los que disponemos, denominado conjunto de entrenamiento. Existen una serie de algoritmos adaptados para este propósito, pero el más extendido es el algoritmo de **Descenso de Gradiente**. Como su propio nombre indica, este se trata de un **método iterativo** que ajusta los parámetros entrenables de una arquitectura (pesos y biases) en la dirección opuesta al gradiente de la función de pérdida con respecto de estos parámetros.

2.3.2.1. Descenso de Gradiente y tasa de aprendizaje

El **gradiente** de la función de pérdida para un conjunto de parámetros provee la dirección en el espacio de parámetros en la cual un pequeño cambio provoca un mayor aumento en la misma, para un conjunto fijo de entrenamiento. La dirección opuesta a este gradiente será, por tanto, la que nos garantiza una disminución más significativa en dicha función. Por ende, si lo que buscamos es minimizar la pérdida de la función que realiza la red neuronal con respecto de las etiquetas verdaderas, será útil modificar los parámetros siguiendo esta dirección opuesta.

Aunque conocemos la dirección de movimiento óptima, resulta complejo determinar qué tanto nos hemos de mover en esa dirección, es decir, el tamaño del paso que hemos de tomar. Aquí es donde entra en juego la **tasa de aprendizaje** λ (*learning rate*), que es un hiperparámetro crítico para ajustar el nivel de aprendizaje de una red neuronal. Así, los parámetros de la red en cada iteración se actualizan de la siguiente manera:

$$\omega^{i+1} = \omega^i - \lambda \cdot \nabla C(\omega^i),$$

donde ω^i representa el vector de parámetros en la iteración $i \in \mathbb{N}$, y C es la función de pérdida empleada, que considera los pares etiquetados de datos. De ser el valor λ demasiado pequeño, el algoritmo tardará muchas más iteraciones de las necesarias en ajustar los parámetros, resultando ineficiente. Por el contrario, si es demasiado grande, el algoritmo podría diverger, por lo que la red no aprendería. En la [Figura 2.2](#), encontramos una interpretación intuitiva de esta idea.

La determinación de la tasa de aprendizaje suele basarse en un proceso heurístico o experimental. Sin embargo, existen métodos avanzados como el **optimizador Adam** (Adaptive

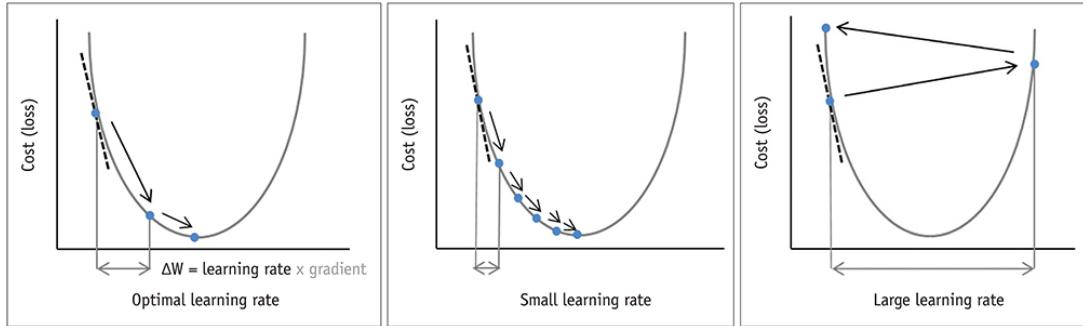


Figura 2.2: Comparación de la convergencia en el entrenamiento de una red neuronal según el método de Descenso de Gradiente, en función de la tasa de aprendizaje. A pesar de que la visualización se reduce al caso de funciones de pérdida unidimensionales, en realidad trabajamos con funciones definidas en \mathbb{R}^p , siendo $p \in \mathbb{N}$ el número total de pesos y biases de la red, que depende del tamaño de la arquitectura entrenada. Imagen extraída de [17].

Moment Estimation) [27], que ajusta dinámicamente la tasa de aprendizaje a lo largo de las iteraciones del algoritmo. Adam utiliza estimaciones de los gradientes anteriores para adaptar la tasa de aprendizaje de cada parámetro individual, mejorando así la convergencia del entrenamiento y siendo menos sensible a la selección inicial de la tasa.

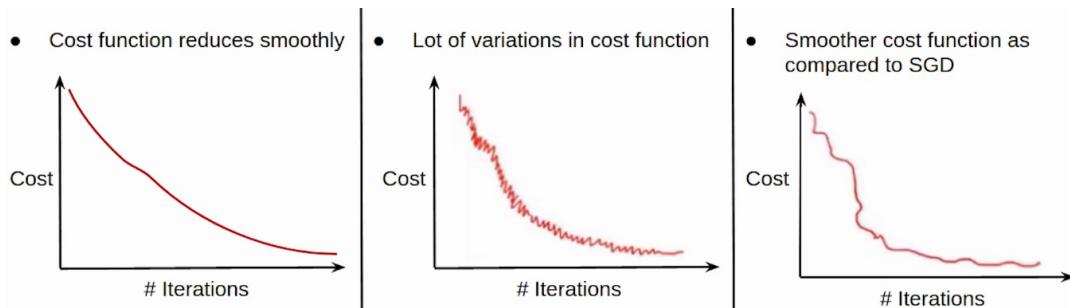
Un aspecto que tenemos que fijar es el valor de los parámetros al comienzo del proceso de entrenamiento, antes de que se realice cualquier actualización sobre los mismos. Por lo general, estos suelen inicializarse como valores **aleatorios** y pequeños que siguen una distribución uniforme o normal, con el objetivo de evitar sesgos en la búsqueda de mínimos de la función de pérdida. Existen otras estrategias más sofisticadas, como algunas de las descritas en [40]. No obstante, ninguna de ellas garantiza que los mínimos encontrados sean globales.

2.3.2.2. Variantes del Descenso de Gradiente

Hay que apuntar que, de utilizar todos los datos del conjunto de entrenamiento al mismo tiempo para el cálculo del gradiente, el gasto de recursos puede ser muy grande. Para reducir este coste, se suele optar por técnicas que no consideran todos ellos a la vez, sino subconjuntos aleatorios de los mismos a los que denominamos **mini-lotes** (o mini-batches). Este enfoque se denomina **Descenso de Gradiente Estocástico con Mini-Lotes** [33]. La idea es calcular el gradiente para cada mini-lote y actualizar los parámetros del modelo en consecuencia, completando así una **iteración**. Una vez se hayan procesado todos los mini-lotes, se habrá usado el conjunto de entrenamiento completo para la actualización de pesos, con lo que habremos finalizado una **época**. Normalmente, se suele barajar el conjunto de mini-lotes antes de comenzar cada época, de nuevo introduciendo una componente aleatoria que ayuda a que el proceso de entrenamiento no esté condicionado.

Cuando se seleccionan tamaños estratégicos (especialmente potencias de dos), los mini-lotes logran sacar el mayor provecho de las GPUs y CPUs modernas, suponiendo una mejora en términos de eficiencia computacional. Estos aspectos se explican con detalle en [41]. Además, el uso de mini-lotes mejora la capacidad de adaptación del modelo a datos que no se usan en su entrenamiento, como se observa en artículos como [63] o [26].

Por otro lado, el método conocido como Descenso de Gradiente Estocástico clásico utiliza un tamaño de mini-lote igual a uno, pero esto puede resultar en una convergencia muy ruidosa debido a la actualización sucesiva de los pesos basándose en una sola instancia de entrenamiento, que puede no ser representativa del conjunto completo. El enfoque de Descenso de Gradiente puro, por su parte, ofrece una estabilidad muy buena, pero no generaliza demasiado bien a datos no visualizados durante el entrenamiento. Por tanto, la alternativa que utiliza mini-lotes supone un balance entre **generalización y estabilidad en la convergencia**, mejorando además la eficiencia computacional. En la [Figura 2.3](#), se observa el comportamiento de cada variante en términos de estabilidad a través de las sucesivas actualizaciones de los parámetros.



[Figura 2.3](#): Comparación de la convergencia en el entrenamiento de una red según la variante de Descenso de Gradiente utilizada. De izquierda a derecha, encontramos gráficas para el Descenso de Gradiente puro, el Descenso de Gradiente Estocástico clásico y el Descenso de Gradiente Estocástico con Mini-Lotes. El eje vertical muestra la pérdida calculada a partir de los datos de entrenamiento y el eje horizontal representa la evolución de las iteraciones (las actualizaciones de los parámetros). Imagen extraída de [54].

2.3.2.3. Propagación hacia atrás

El método de Descenso de Gradiente nos ofrece unas instrucciones para minimizar la función de pérdida usando los gradientes, pero no nos indica una forma efectiva de calcularlos. En nuestro estudio lidiamos con redes totalmente conectadas, cuya forma hemos analizado con detalle en la [Sección 2.2](#). Como las únicas conexiones en este tipo de red son de cada capa hacia la siguiente, podemos establecer un método para **calcular los gradientes** de manera eficiente utilizando la técnica de propagación hacia atrás (o *backpropagation*, en inglés) [49].

Esta técnica implica dos fases: la fase hacia adelante y la fase hacia atrás.

- En la **fase hacia adelante**, la entrada se propaga a través de la red capa por capa para generar una predicción. Esta predicción se compara con la salida deseada utilizando la función de pérdida correspondiente.
- En la **fase hacia atrás**, el error calculado se propaga hacia atrás a través de la red. Comenzando desde la capa de salida, se calcula el gradiente del error con respecto a los pesos utilizando la regla de la cadena. Dado que cada capa de estas redes solo está conectada con la siguiente, las derivadas parciales de los parámetros en una capa se pueden reutilizar en el cálculo de las de la capa anterior. Este flujo inverso de la

información del error permite una computación eficiente del gradiente completo, en contraste con el enfoque de irlo calculando en cada capa por separado, que resulta altamente costoso. Los pesos se van actualizando a medida que retrocedemos hacia atrás en la arquitectura usando, ahora sí, el método de Descenso de Gradiente.

La formulación matemática de este procedimiento no es complicada, pero resulta bastante tediosa en términos de notación. Para consultarla en detalle, se recomiendan artículos como [25] o [14]. A nivel divulgativo, también resulta de interés el recurso [3].

2.3.2.4. Adaptación de las funciones de activación al entrenamiento

Las funciones de activación expuestas en la Subsección 2.1.4 se utilizan para introducir no linealidades (*non-linearities*) en las arquitecturas neuronales con la intención de hacerlas más expresivas. Con el desarrollo del algoritmo de propagación hacia atrás en la década de 1980, se empezó a popularizar la función de activación **sigmoide**. Esta función es la base del primer Teorema de Aproximación para redes neuronales, que estudiaremos en la Sección 3.1. Sin embargo, en lo que respecta al entrenamiento, esta función de activación presenta dos importantes inconvenientes:

- **Ineficiencia computacional:** La función sigmoide involucra una operación exponencial. Esta operación consume muchos recursos, tanto en la inferencia del modelo como en la computación de los gradientes. El resultado es un entrenamiento muy lento, especialmente para arquitecturas de gran tamaño.
- **Desvanecimiento del gradiente** (*vanishing gradients*): La derivada de la función sigmoide es muy baja para valores de entrada que están a suficiente distancia del cero, como se deduce de su fórmula (2.2). En consecuencia, puede ocurrir que las derivadas parciales calculadas por el método de propagación hacia atrás en cada capa disminuyan cada vez más su magnitud durante la fase hacia atrás, de manera que apenas se actualizan los parámetros en las primeras capas, impidiendo el modelado efectivo de patrones en ellas. Evidentemente, este hecho se agrava cuanto más profunda sea la red entrenada. En consecuencia, su poder de representación puede verse comprometido.

Otras funciones de activación similares, como la tangente hiperbólica, presentan limitaciones parecidas. Para abordar estos obstáculos en el entrenamiento de redes neuronales, se introdujo la función de activación **ReLU**, que de hecho resuelve estos dos problemas: ReLU no se satura para valores positivos, puesto que su derivada es siempre uno en este dominio, lo que permite la propagación hacia atrás sin pérdidas. Además, su forma es extremadamente sencilla (se puede implementar mediante una simple sentencia *if*), por lo que su manejo es poco costoso. En las arquitecturas neuronales modernas, es la función de activación más extendida, junto con sus variantes.

Sin embargo, ReLU presenta una problemática: la **muerte de neuronas** (*dying ReLU*) [35]. Este problema ocurre cuando una neurona ReLU recibe constantemente entradas negativas, llevando su salida a cero y, como resultado, su gradiente también a cero. A partir de ese momento, la neurona no contribuirá a la salida ni al entrenamiento del modelo, siendo este efecto irreversible. En consecuencia, la neurona queda inservible, por lo que el modelo puede perder expresividad, aunque no en la misma medida que en el caso del desvanecimiento del gradiente, que afecta a capas completas cercanas a la entrada.

Igualmente, este inconveniente se puede salvar usando la variante **Leaky ReLU** [64], que introduce un pequeño gradiente para entradas negativas, de forma que las neuronas sigan aprendiendo incluso cuando reciben entradas negativas constantes, lo cual aumenta la robustez del modelo. Hay otras alternativas parecidas, como ALReLU [38] o RReLU [9].

2.3.3. Generalización en redes neuronales

Como ya se menciona en la [Subsección 2.3.2](#), no es suficiente con que el entrenamiento de una red neuronal garantice un buen ajuste para unos datos concretos, sino que debe constituir un modelo que se adapte adecuadamente a datos del mismo problema que no se hayan considerado durante el entrenamiento, de manera que podamos usarlo para realizar predicciones sólidas sobre nuevas entradas. Por tanto, se han de proveer herramientas para medir el nivel de generalización de un modelo, y para mejorarlo en su caso.

2.3.3.1. Métricas principales

En primer lugar, es fundamental utilizar métricas adecuadas que permitan cuantificar el desempeño de un modelo para un conjunto de datos. Se dividen según el tipo de problema:

- **Métricas para clasificación:** La métrica más sencilla en problemas de clasificación es la **exactitud** (*accuracy*), que se calcula como la proporción de predicciones correctas sobre el total de predicciones realizadas. Es una medida representativa siempre y cuando las clases estén balanceadas en el conjunto que se predice.

En problemas de clasificación binaria, al solo haber dos etiquetas, podemos identificar una como caso positivo y otra como caso negativo. Entonces, podemos definir las siguientes categorías:

- **Verdaderos Positivos (TP):** Instancias correctamente clasificadas como positivas. Por ejemplo, diagnosticar adecuadamente a un paciente enfermo.
- **Falsos Positivos (FP):** Instancias incorrectamente clasificadas como positivas. Un caso sería diagnosticar a un paciente saludable como enfermo.
- **Falsos Negativos (FN):** Instancias incorrectamente clasificadas como negativas. Por ejemplo, considerar erróneamente que un paciente enfermo está sano.
- **Verdaderos Negativos (TN):** Instancias correctamente clasificadas como negativas. Un ejemplo sería confirmar que un paciente sano no está enfermo.

En base a estas categorías, se plantean las siguientes métricas para clasificación binaria:

- **Precisión:** Es la proporción de verdaderos positivos (TP) sobre el total de positivos predichos (TP + FP). En el contexto del diagnóstico de enfermedades, mide la proporción de pacientes diagnosticados como enfermos que realmente están enfermos.
- **Sensibilidad (*recall*):** Es la proporción de verdaderos positivos (TP) sobre el total de positivos reales (TP + FN). En el mismo ámbito, mide qué tantos pacientes enfermos hemos identificado correctamente de entre todos los que lo están.
- **F1-Score:** Es la media armónica de la precisión y la sensibilidad, proporcionando un equilibrio entre ambas métricas.

$$F1 = 2 \cdot \frac{\text{Precisión} \cdot \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}}$$

Para extender estas métricas al caso de **clasificación multiclasa**, se pueden calcular promedios ponderados. Esto se hace calculando individualmente las métricas anteriores considerando como caso positivo una de las clases y como negativo el resto de ellas, y realizando una suma ponderada de todos los valores teniendo en cuenta la proporción de instancias de cada clase en el conjunto de datos. Así, se obtiene una visión general del rendimiento del modelo en todas las clases.

Por último, destacamos la **matriz de confusión**, que es una tabla que muestra el número de predicciones correctas e incorrectas desglosadas por cada clase. También nos ayuda a detectar cuáles son las clases que se están confundiendo más entre ellas.

- **Métricas para regresión:** Se suele utilizar el propio **Error Cuadrático Medio** (MSE), cuya fórmula está definida en (2.4) para medir el rendimiento de modelos de regresión, al ser una medida muy interpretable. Una variante del MSE es la **Raíz del Error Cuadrático Medio** (RMSE), que devuelve la raíz cuadrada del MSE y mantiene las mismas unidades que la variable objetivo, facilitando aún más la interpretación de los errores. Otra métrica común es el **Error Absoluto Medio** (MAE), que mide la media de los errores absolutos y es menos sensible a los valores atípicos comparado con el MSE.

Destacaremos además el **Coeficiente de Determinación** (R^2), que es una medida de la proporción de la varianza en la variable dependiente que es predecible a partir de la variable independiente. Proporciona una indicación de la calidad del ajuste del modelo. Su fórmula es

$$R^2 = 1 - \frac{\sum_{i=1}^k (f(x_i) - \mathcal{A}(x_i))^2}{\sum_{i=1}^k (f(x_i) - \bar{y})^2},$$

donde $\{x_i\}_{i=1}^k$, $k \in \mathbb{N}$, es el conjunto de entradas que queremos predecir mediante la red \mathcal{A} , f es la función que representa el modelo (toma inputs y produce etiquetas verdaderas), y $\bar{y} = \frac{1}{k} \sum_{i=1}^k f(x_i)$, es decir, la media de todas las etiquetas. Un valor R^2 cercano a uno indica que una gran proporción de la varianza en la variable dependiente ha sido explicada por el modelo. Un R^2 cercano a cero sugiere que el modelo no explica bien la variabilidad de la variable dependiente. Un valor negativo de R^2 puede ocurrir cuando el modelo es inapropiado y su ajuste es peor que el que toma simplemente una media de las etiquetas.

2.3.3.2. Partición de los datos etiquetados

Para buscar que el modelo neuronal construido tenga un comportamiento deseable, se suele dividir inicialmente el conjunto de datos etiquetados disponibles en tres subconjuntos:

- **Conjunto de entrenamiento:** Serán los datos involucrados en el proceso de entrenamiento, es decir, aquellos que se consideran en la función de pérdida que se minimiza mediante el Descenso de Gradiente.
- **Conjunto de validación:** Se trata de un conjunto que se utiliza para evaluar el rendimiento de una arquitectura neuronal sobre datos no vistos a lo largo de su entrenamiento, usando las métricas anteriores. No afecta directamente a las actualizaciones de pesos de la red, pero sí que influye en las decisiones que tomemos a la hora de ajustar los hiperparámetros relativos al propio proceso de entrenamiento, o a la selección de otra arquitectura en su caso.

- **Conjunto de evaluación:** También conocido como conjunto de test o de prueba, consiste en una proporción del conjunto inicial de datos que se reserva exclusivamente para evaluar las predicciones de un modelo neuronal una vez finalizada la fase de aprendizaje, que ha sido convenientemente ajustada según los resultados ofrecidos por los dos anteriores conjuntos. En ningún caso debe influir, aunque sea mínimamente, en el reajuste de parámetros o en las decisiones metodológicas tomadas por el programador: se trata de un conjunto independiente que establece unas medidas finales del desempeño de un modelo.

En cuanto al **balance de la partición**, es crucial que la distribución de los datos en cada subconjunto sea representativa del conjunto original. Dada una batería suficientemente grande de datos, la partición se podría establecer de forma aleatoria, pero en casos donde hay menos datos, se debe ser más cuidadoso. Una práctica común es asignar aproximadamente el 70 % de los datos al conjunto de entrenamiento, el 15 % al conjunto de validación y el 15 % al conjunto de evaluación, aunque estos porcentajes pueden variar según la cantidad total de datos y las necesidades específicas del modelo. Así mismo, se deben descartar **datos atípicos**, es decir, aquellos que detectemos que no siguen un patrón coherente, ya que podrían entorpecer la convergencia hacia parámetros óptimos. También es estándar aplicar un **preprocesado** concreto a los datos, transformándolos en valores numéricos procesables mediante la red, y normalizándolos para asegurar que todos los atributos estén en una escala comparable.

Es importante subrayar que hay que respetar rigurosamente el papel y la independencia de cada uno de estos conjuntos de datos. Un ejemplo sutil de una mala práctica sería, por ejemplo, aplicar un preprocesado a los datos en el que estos se normalizan considerando también los valores presentes en el conjunto de validación y/o evaluación. Esto es incorrecto, pues estamos contaminando los datos de entrenamiento con información externa, y estamos sesgando (aunque sea ligeramente) la distribución de los mismos, afectando en consecuencia al proceso de optimización de parámetros. Por tanto, podríamos obtener a posteriori una mala intuición de la capacidad real de generalización del modelo. Estaríamos pues ante un escenario de *data snooping* [62].

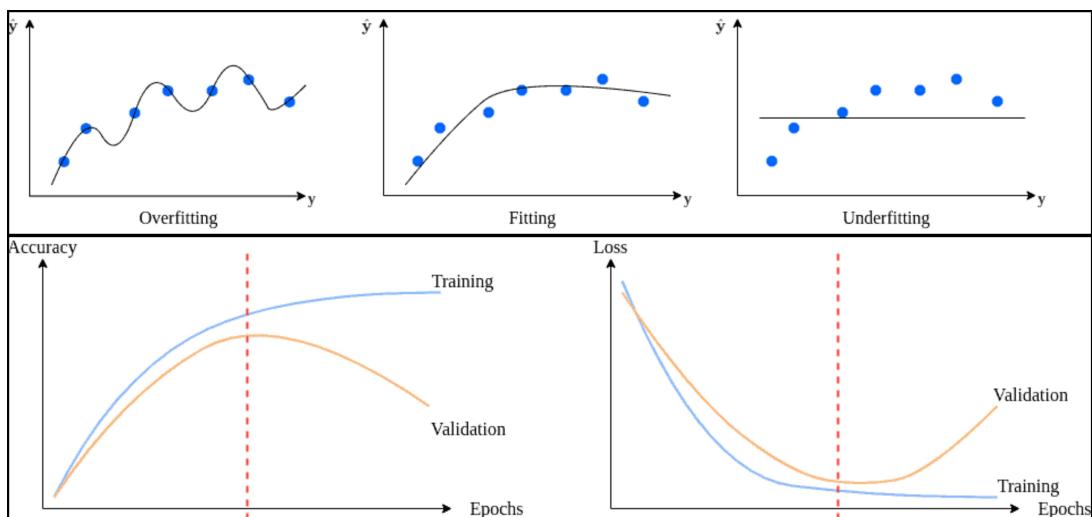
Existen enfoques que buscan mejorar la fiabilidad de las métricas de rendimiento que podemos obtener durante el entrenamiento. Uno de los más utilizados es la **validación cruzada** (*cross-validation*) [12], que primero realiza una partición en dos conjuntos, uno de los cuales se establece como conjunto de test, y el otro se divide en un número m de pliegues. El modelo se entrena m veces tomando en cada ocasión un pliegue como conjunto de validación y el resto de ellos como conjunto de entrenamiento. Se promedian las métricas de todos los entrenamientos, lo cual nos proporciona una visión más general de la capacidad de generalización del modelo. Tras haber elegido una arquitectura y una metodología de entrenamiento definitivas, se entrenaría la red neuronal utilizando todos los pliegues obtenidos, pudiendo entonces aplicar el test.

2.3.3.3. Curvas de aprendizaje

Hemos visto que la generalización de un modelo se mide usando las métricas aplicadas sobre el conjunto de validación durante el entrenamiento. La forma exacta de hacerlo pasa por estudiar las **curvas de aprendizaje**, que muestran conjuntamente la evolución de una métrica concreta para el conjunto de entrenamiento y el de validación a través de las sucesivas épocas. Al analizarlas, podemos detectar dos fenómenos:

- **Infraajuste (underfitting):** Sucede cuando el modelo tiene dificultades para predecir los datos de entrenamiento. Este fenómeno suele ocurrir cuando no existe un equilibrio entre el tamaño de la arquitectura y el número de datos y/o su dificultad, o bien cuando entrenamos el modelo durante pocas épocas o con hiperparámetros de aprendizaje incorrectos (por ejemplo, una tasa de aprendizaje demasiado alta).
- **Sobreajuste (overfitting):** En este caso, el modelo se ajusta excesivamente bien a los datos de entrenamiento, pero no tanto así al conjunto de validación. Precisamente, las arquitecturas neuronales con muchos parámetros o aquellas que se entran con un conjunto escaso de datos son propensas a sufrir este problema. Por otro lado, si entramos demasiadas épocas, el modelo puede empezar a ser demasiado específico a los datos de entrenamiento, lo cual suele conllevar una generalización pobre a nuevas entradas.

Por tanto, el objetivo es encontrar un compromiso entre el ajuste de los datos de entrenamiento y la generalización a nuevas muestras, es decir, un **ajuste óptimo**. En la parte inferior de la [Figura 2.4](#), se muestra el aspecto de las curvas de aprendizaje en cada caso. Como vemos, se pueden graficar las curvas con respecto de varias métricas, y se han de examinar notando si el objetivo es maximizar o minimizar la métrica concreta.



[Figura 2.4:](#) En la parte superior de la figura, observamos una representación de la función que realiza una red según su tipo de ajuste (de izquierda a derecha: sobreajuste, ajuste óptimo e infraajuste). En la parte inferior, observamos a la izquierda la evolución de una métrica de precisión (se ha de maximizar) y a la derecha una de pérdida (se ha de minimizar). En cada caso, a la izquierda de la línea discontinua roja, el modelo está infraajustado, y a su derecha empieza a sobreajustarse. En la época marcada por la línea, el ajuste es óptimo. Imagen extraída de [2].

El análisis de la evolución de las métricas resulta necesario para determinar el nivel de aprendizaje de un modelo, e intentar mejorarlo en caso de necesitarlo, modificando los hiperparámetros de entrenamiento y/o incorporando técnicas como las que veremos en la [Subsubsección 2.3.3.4](#). No nos basta con examinar las métricas en la última época: debemos estudiar el progreso de las mismas durante el entrenamiento.

Si tras sucesivos intentos de mejoría en el proceso de aprendizaje estas curvas aún no tienen un buen comportamiento, es posible que simplemente la arquitectura no sea indicada para el problema o conjunto de datos concreto, en cuyo caso habría que descartarla y probar con otra. La evaluación de métricas sobre el conjunto de test solo se realiza sobre el modelo cuyo rendimiento de aprendizaje consideremos óptimo.

2.3.3.4. Estrategias para la generalización

A continuación, se describen algunas prácticas adicionales que se llevan a cabo para tratar de mejorar la generalización de un modelo neuronal, y cuyo efecto se podrá comprobar examinando las curvas de aprendizaje.

- **Detención temprana** (*early stopping*): Se implementa de forma que el entrenamiento de un modelo se detenga automáticamente cuando el conjunto de datos de validación lleva unas cuantas épocas sin mejorar sus métricas en demasía. El umbral se establece experimentalmente y dependiendo del caso. Así, se previene el sobreajuste del modelo.
- **Regularización:** Implica añadir un término de penalización a la función de pérdida para los parámetros del modelo. Las formas más comunes son la regularización L_1 (Lasso) y L_2 (Ridge), que ayudan a evitar que el modelo se vuelva demasiado complejo. Sus ventajas y desventajas se comentan en trabajos como [42].
- **Dropout:** Es una técnica que fue propuesta en propuesta en [55], y que consiste en desactivar aleatoriamente un porcentaje de neuronas durante el entrenamiento en cada paso de actualización, lo que ayuda a prevenir el sobreajuste al reducir la dependencia del modelo en neuronas específicas.

La combinación de algunos de estos métodos con un **Descenso de Gradiente Estocástico con Mini-Lotes** y el **optimizador Adam**, descritos ambos en [Subsección 2.3.2](#), suelen ayudar a mejorar la generalización, aunque hay que estudiar el caso concreto y tener en cuenta que muchas veces es la propia arquitectura y su excesiva simplicidad (o complejidad) la que está impidiendo el buen desempeño del modelo para datos nuevos.

3 Capacidad de Cómputo de las Redes Neuronales

En el estudio de las redes neuronales, uno de los objetivos fundamentales es entender su potencial para **aproximar funciones** complejas. Esta capacidad es crucial tanto desde una perspectiva teórica como práctica, pues determina hasta qué punto una red neuronal puede aprender y generalizar a partir de datos.

Una de las vías para comprobar esta capacidad son las **pruebas existenciales**, que simplemente demuestran si una cierta propiedad es posible, estableciendo limitaciones teóricas en algunos casos. Por otro lado, también existen **pruebas constructivas**, que no solo demuestran la existencia de dicha propiedad, sino que también describen cómo construir ejemplos específicos que la posean.

En la práctica del aprendizaje automático, las pruebas constructivas nos proporcionan un posible un camino para el diseño de modelos de mayor explicabilidad. Exploraremos con mayor detalle este camino en el [Capítulo 4](#).

3.1. Redes acotadas en profundidad

A continuación, presentamos un resultado teórico clásico en el campo de las redes neuronales. Este fue demostrado por George Cybenko en el artículo [\[13\]](#) de 1989, y nos ofrece una buena intuición general de la capacidad de cómputo de las arquitecturas neuronales totalmente conectadas. En concreto, se prueba que una sola capa oculta es suficiente para aproximar cualquier función real continua definida en $[0, 1]^n$. Eso sí, dependiendo de la función concreta y de la precisión requerida, la anchura de la red podría ser arbitrariamente grande, lo cual supone un reto de cara al ámbito práctico.

Teorema 3.1 (G. Cybenko). *Sea una función $f : [0, 1]^n \rightarrow \mathbb{R}$ continua y $\epsilon > 0$, existe una red neuronal totalmente conectada, con una única capa oculta, que realiza una función $g : \mathbb{R}^n \rightarrow \mathbb{R}$ tal que*

$$|f(x) - g(x)| < \epsilon \quad \forall x \in [0, 1]^n.$$

Observación 3.1.1. Concretamente, la función g indicada tiene la siguiente forma:

$$g(x) = \sum_{j=1}^m \mathcal{W}_{1,j}^{(2)} \cdot \sigma \left(\sum_{i=1}^n \mathcal{W}_{j,i}^{(1)} \cdot x_i - \mathcal{B}_j^{(1)} \right) \quad \forall x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n,$$

donde $\mathcal{W}^{(k)}$ es la matriz de pesos para la capa $k \in \{1, 2\}$, $\mathcal{B}^{(1)}$ es el vector de biases de la primera capa, y $m \in \mathbb{N}$ un valor arbitrario. Nótese que no se incluye bias para calcular el valor de salida.

En cuanto a σ , basta con que sea una función continua en \mathbb{R} y de aplastamiento, es decir, que verifique la condición dada por [\(2.1\)](#). Sin embargo, por comodidad, realizaremos la exposición con la función sigmoide, dada por $\sigma(x) = \frac{1}{1+e^{-x}}$ para cada $x \in \mathbb{R}$, que se utiliza ampliamente en este contexto.

La demostración rigurosa del [Teorema 3.1](#), que se encuentra detallada en [13], implica el manejo de resultados de Análisis Funcional como Teorema de Hahn-Banach [39], el Teorema de Representación de Riesz [46] o la Transformada de Fourier [52]. Se trata de una demostración meramente existencial, pero no la abordaremos desde ese punto de vista, sino tratando de proveer una idea intuitiva de cómo se construiría una red de estas características de forma explícita. Nos centraremos en el caso de las funciones reales continuas de una sola variable, de tal manera que la red neuronal asociada solo tomará un valor de entrada.

En primer lugar, dado un $\delta > 0$, planteamos una red neuronal Φ_δ , de una sola capa oculta y con función de activación sigmoide. Esta red debe ser tal que la gráfica $\mathcal{G}(\Phi_\delta)$ describa una silueta similar a la de la función $\mathcal{G}(h \cdot \chi_I)$ siendo I un intervalo de \mathbb{R} con extremos $a, b \in \mathbb{R}$, $a < b$, y $h \in \mathbb{R}$. Entonces, definimos la función realizada por la red como

$$\Phi_\delta(x) = h \cdot \sigma\left(\frac{x-a}{\delta}\right) - h \cdot \sigma\left(\frac{x-b}{\delta}\right) \quad \forall x \in \mathbb{R}, \quad (3.1)$$

lo cual es posible tomando $\mathcal{W}_{1,1}^{(1)} = \mathcal{W}_{2,1}^{(1)} = \frac{1}{\delta}$, $\mathcal{B}_1^{(1)} = \frac{a}{\delta}$, $\mathcal{B}_2^{(1)} = \frac{b}{\delta}$, y $\mathcal{W}_{1,1}^{(2)} = -\mathcal{W}_{1,2}^{(2)} = h$, tal como se refleja en la arquitectura situada a la izquierda en la [Figura 3.1](#). Como se indicaba, todas las neuronas de la capa oculta tienen función de activación sigmoide, y la salida se calcula de forma lineal, sin usar el término de bias.

Analicemos algunas propiedades de esta función, las cuales son fácilmente demostrables usando las propiedades de continuidad, crecimiento y simetría de la sigmoide vistas en la [Subsección 2.1.4](#).

Propiedad 3.1.1. Φ_δ es continua y derivable en \mathbb{R} .

Propiedad 3.1.2. La imagen de Φ_δ en \mathbb{R} está acotada entre 0 y h . Esto es, $\Phi_\delta(\mathbb{R}) \subset]0, h[$.

Propiedad 3.1.3. Φ_δ es simétrica con respecto del eje $x = \frac{a+b}{2}$.

Propiedad 3.1.4. Φ_δ crece estrictamente en $]-\infty, \frac{a+b}{2}[$ y decrece estrictamente en $]\frac{a+b}{2}, +\infty[$.

Propiedad 3.1.5. Cuando $x \in \mathbb{R} \setminus \{a, b\}$, se tiene que

$$\lim_{\delta \rightarrow 0} \Phi_\delta(x) = h \cdot \chi_I(x).$$

Además, $\lim_{\delta \rightarrow 0} \Phi_\delta(a) = \lim_{\delta \rightarrow 0} \Phi_\delta(b) = \frac{1}{2}$.

Propiedad 3.1.6. Dados $\epsilon > 0$ y $\gamma > 0$, podemos tomar un $\delta > 0$ tal que

$$|\Phi_\delta(x) - h \cdot \chi_I(x)| < \epsilon \quad \forall x \in \mathbb{R} \setminus ([a - \gamma, a + \gamma] \cup [b - \gamma, b + \gamma])$$

La [Propiedad 3.1.6](#) nos indica que, si bien siempre podremos garantizar la acotación uniforme arbitraria del error de Φ_δ con respecto de $h \cdot \chi_I$ en subconjuntos de \mathbb{R} cada vez más grandes tomando cierto $\delta > 0$, siempre existirán dos ventanas determinadas por $[a - \gamma, a + \gamma]$ y $[b - \gamma, b + \gamma]$ donde el error no se puede controlar por debajo de un ϵ . Estas ventanas pueden hacerse tan pequeñas como se deseé.

Este argumento también deja claro que, si tomamos un intervalo $[c, d]$, con $c, d \in \mathbb{R}$, $c < d$, tal que $[a, b] \subset [c, d]$, entonces

$$\begin{aligned} \int_{[c,d]} |\Phi_\delta - h \cdot \chi_I| &< \int_{[c,a]} \epsilon + \int_{[a-\gamma, a+\gamma]} |h| + \int_{[a-\gamma, b-\gamma]} \epsilon + \int_{[b-\gamma, b+\gamma]} |h| + \int_{[b,d]} \epsilon \\ &< \epsilon \cdot (d - c) + 4\gamma \cdot h < \infty. \end{aligned}$$

Eligiendo ϵ y γ suficientemente pequeños, podríamos minimizar la distancia entre Φ_δ y $h \cdot \chi_I$ en $L^1([c, d])$. Todos los comportamientos descritos se reflejan en la parte derecha de la Figura 3.1.

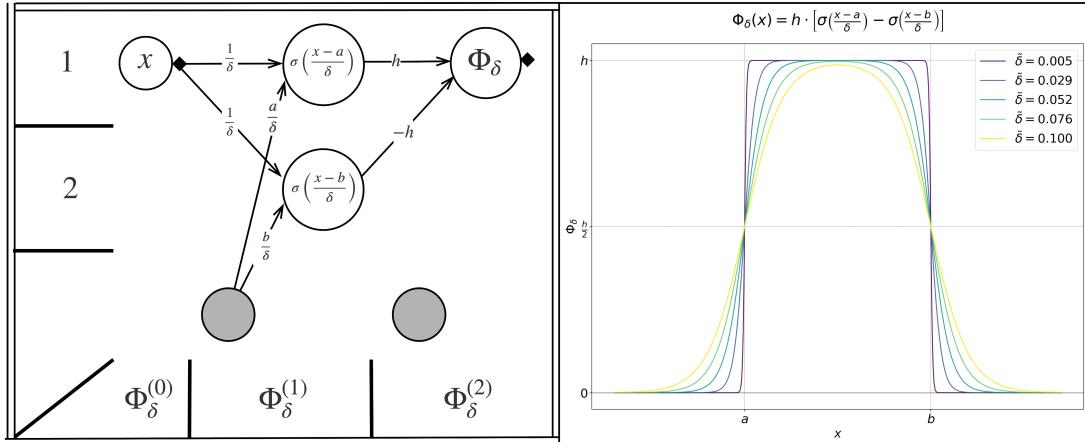


Figura 3.1: A la izquierda se muestra un esquema de la arquitectura de la red neuronal Φ_δ cuya expresión figura en (3.1), mientras que a la derecha se muestra la evolución de la forma de la función que realiza Φ_δ a medida que δ se aproxima a cero. El término $\tilde{\delta}$ que se indica en la leyenda de la figura es tal que $\delta = \tilde{\delta} \cdot (b - a)$. Al normalizar δ en función de la longitud del intervalo I de extremos a y b , aseguramos que la disposición relativa de las gráficas no se altera al modificar dicha longitud. Suponemos $h > 0$.

Observación 3.1.2. Para el caso n -dimensional, con $n > 1$, también es posible diseñar una red neuronal con una única capa oculta y funciones de activación sigmoides, la cual es capaz de realizar una función cuya gráfica se asimile arbitrariamente a la de la función $h \cdot \chi_I$, siendo I cualquier intervalo de \mathbb{R}^n y $h \in \mathbb{R}$. El argumento, pese a que también es visual, resulta un poco más tedioso. Además, requiere de un incremento sustancial en la anchura de la red. Se recomienda consultar referencias como [43] o [11] para conocer más detalles.

Una vez que conocemos esta arquitectura neuronal, podemos extenderla para que pueda aproximar funciones reales continuas en $I = [a, b]$. En virtud de la Proposición 1.7.1, y sabiendo que I es compacto, podemos afirmar que realizando una partición suficientemente fina del intervalo I , podremos construir una combinación lineal finita de funciones características sobre los elementos de esa partición de forma que aproximemos $f(x)$ con un grado de precisión uniforme $\forall x \in I$, siendo $f : I \rightarrow \mathbb{R}$ una función continua. De manera más informal, este tipo de funciones pueden ser aproximadas mediante funciones escalonadas, acotando el margen de error en la medida que deseemos. En principio, no tenemos certeza de cuántos conjuntos necesitamos en la partición para una función y una precisión arbitrarias.

3 Capacidad de Cómputo de las Redes Neuronales

Utilizando este argumento, podemos tratar de plantear una red neuronal g que compute una función continua $f : I \rightarrow \mathbb{R}$ de la siguiente manera:

- Definimos $I_j = [a + \frac{(b-a) \cdot (j-1)}{k}, a + \frac{(b-a) \cdot j}{k}]$, $j \in \Gamma_{k-1}$; e $I_k = [a + \frac{(b-a) \cdot (k-1)}{k}, b]$. Sabemos que podemos escoger $k \in \mathbb{N}$ de forma que garanticemos que f tiene una variación menor que cierto $\epsilon > 0$ en cada subintervalo de la partición $\{I_j\}_{j=1}^k$ de $[a, b]$.
- Extendemos la capa oculta de la arquitectura vista en la [Figura 3.1](#) de forma que se incluyan dos neuronas como las que se muestran para cada subintervalo. Concretamente, dado un $j \in \Gamma_k$, incluiremos neuronas $g_{2j-1}^{(1)}$ y $g_{2j}^{(1)}$, que recibirán el valor de entrada mediante las conexiones dadas por los pesos $W_{2j-1,1}^{(1)} = W_{2j,1}^{(1)} = \frac{1}{\delta}$, donde $\delta > 0$ es el factor de aproximación que discutimos anteriormente. Además, los respectivos biases serán $B_{2j-1}^{(1)} = \frac{a}{\delta} + \frac{(b-a) \cdot (j-1)}{k \cdot \delta}$ y $B_{2j}^{(1)} = \frac{a}{\delta} + \frac{(b-a) \cdot j}{k \cdot \delta}$. En cuanto a las conexiones salientes, incluimos $W_{1,2j-1}^{(2)} = f(c_j)$ y $W_{1,2j}^{(2)} = -f(c_j)$, donde c_j es un punto del intervalo I_j . Podemos seleccionar, por ejemplo, el punto intermedio $c_j = a + \frac{(b-a) \cdot (2j-1)}{2k}$. El resto de conexiones de la red son nulas, es decir, los pesos son igual a cero.
- De esta forma, para un input $x \in [0, 1]$, la red g suma las contribuciones de las neuronas que aproximan la función característica de cada subintervalo, de forma que el valor de salida $g(x) = \sum_{j=1}^k f(c_j) (g_{2j-1}^{(1)}(x) - g_{2j}^{(1)}(x))$ sea muy cercano a $f(c_j)$ para $x \in I_j$, $j \in \Gamma_k$.

En la [Figura 3.2](#), se muestra un ejemplo de la arquitectura recién descrita cuando dividimos el intervalo $[a, b]$ en $k = 3$ subintervalos uniformemente distribuidos. Así mismo, se muestran gráficas de la salida de g , con distintos valores de δ .

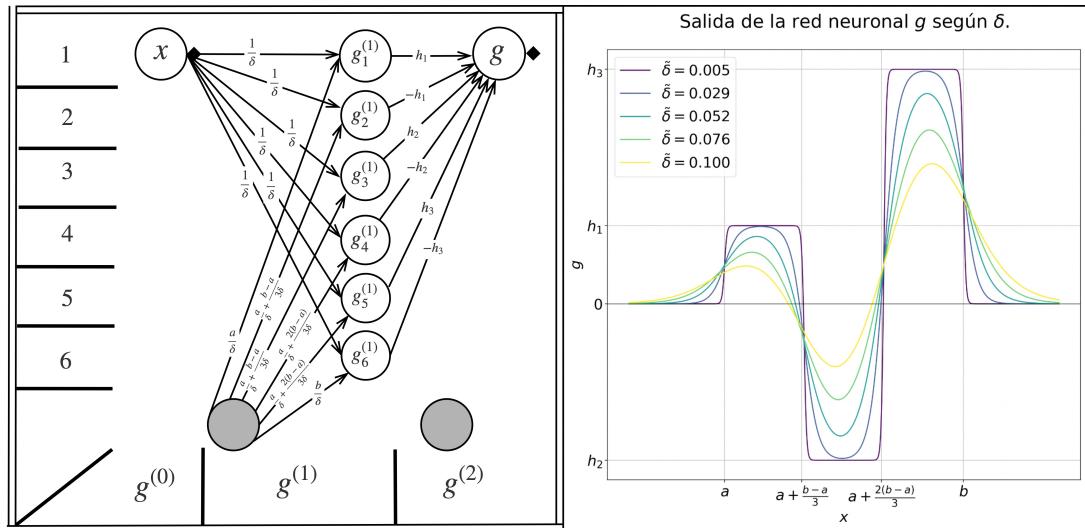


Figura 3.2: A la izquierda se muestra un ejemplo de la arquitectura propuesta para aproximar funciones continuas en $[a, b]$, con $k = 3$. A la derecha, se muestran gráficas de su función de salida según el valor δ fijado durante su construcción. Se toma $\tilde{\delta}$ como en la [Figura 3.1](#). Dado $h > 0$, suponemos $h_1 = h$, $h_2 = -2h$ y $h_3 = 3h$.

Observamos que si δ es demasiado grande, para cada $x \in [a, b]$ no solo arrastramos el error de la aproximación de la función característica del subintervalo en que se encuentra x , sino también el que producen las neuronas que aproximan las funciones características del resto de subintervalos. Es por eso que δ debe ser suficientemente pequeño, asegurando una aproximación razonable de cada punto y la acumulación del menor error posible.

Aún así, por lo expuesto en la [Propiedad 3.1.6](#), siempre va a haber ventanas alrededor de cada extremo de un subintervalo en donde no podemos acotar el error. Para solventar esta problemática, una solución podría ser desfasar los intervalos iniciales de la partición un número d de veces, e incluir neuronas para aproximar las funciones características de estos nuevos subintervalos desfasados en la capa oculta. Podemos así tratar de que cada punto quede en una única ventana de error. Realizando suficientes desfases, y multiplicando las conexiones con la salida por $\frac{1}{d}$, podremos asegurar que todos los puntos quedan a una distancia menor que un $\epsilon > 0$ dado de la función inicial que se quería aproximar. Sin embargo, en la práctica, nos conformaremos con que las ventanas de error sean mínimas, ajustando el valor de δ convenientemente.

Como cabe esperar por el enunciado de la [Proposición 1.7.1](#), este argumento es igualmente válido para funciones continuas definidas en cualquier compacto de \mathbb{R}^n , con $n \in \mathbb{N}$. De hecho, en el artículo [\[24\]](#), se incluye el siguiente resultado:

Teorema 3.2 (Hornik-Stinchcombe-White). *La clase de las funciones realizadas por redes neuronales de una única capa oculta y con funciones de activación de aplastamiento, es uniformemente densa en compactos con respecto a la clase de las funciones continuas.*

Esto es, no hay ninguna restricción para extender el resultado de Cybenko visto en el [Teorema 3.1](#) a funciones continuas definidas en compactos arbitrarios de \mathbb{R}^n . Esta clase, de hecho, contiene a la mayoría de funciones de interés que nos interesarán modelar en el ámbito de aplicación de las redes neuronales.

Además, en la referencia [\[32\]](#), se prueba que ni siquiera es necesario que las funciones de activación de la capa oculta sean de aplastamiento, simplemente han de ser no polinomiales. Las restricciones sobre las funciones de activación en redes de una sola capa oculta son por tanto mínimas para lograr la aproximación universal.

3.2. Redes acotadas en anchura

Los resultados anteriores, pioneros en la búsqueda de teoremas de aproximación universal para redes neuronales, tienen un gran peso histórico en este campo. Sin embargo, el avance en esta materia ha dado lugar a estudios igualmente valiosos y relevantes.

A continuación, presentamos una serie de resultados que podemos encontrar en [\[37\]](#). Estos siguen la misma línea de investigación, pero añaden tres elementos adicionales:

- Uso de funciones de activación **ReLU**, que junto con sus variantes, son las más utilizadas en las arquitecturas modernas.
- Planteamiento del concepto de aproximación mediante la **distancia en \mathcal{L}^1** , abriendo así la posibilidad a extender los resultados de aproximación universal a funciones más allá de las continuas.
- **Acotación de la anchura** de las redes, explorando así un factor menos recurrente en este ámbito, y tratando de identificar su rol en una arquitectura neuronal.

Haremos especial énfasis en el primer teorema del artículo, al que haremos referencia como *Teorema de Aproximación Universal para redes ReLU limitadas por anchura*. Proveeremos, de manera constructiva, todos los elementos necesarios para demostrarlo, haciendo uso de los fundamentos matemáticos introducidos.

3.2.1. Aproximación de la función característica de un cubo

Lema 3.1. *Sea C un cubo de \mathbb{R}^n contenido en $K := [-N, N]^n$ para cierto $N > 0$. Entonces, existe una red neuronal ReLU totalmente conectada de anchura $w = n + 2$ y profundidad $d = 4n$ que realiza una función $\Phi \in \mathcal{L}^1$ de forma que, dado un $\epsilon > 0$,*

$$\|\chi_C - \Phi\|_{\mathcal{L}^1} < \epsilon.$$

Demostración. Vamos a construir una red neuronal ReLU totalmente conectada que realiza una función $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}$, con la siguiente estructura:

- La anchura total de la red será $w = n + 2$ y su profundidad, $d = 4n$.

Podemos escribir $\Phi = \Phi^{(4n)} \circ \Phi^{(4n-1)} \circ \dots \circ \Phi^{(1)}$, donde para $k \in \Gamma_{4n}$, $\Phi^{(k)}$ es la función que realiza la k -ésima capa de la red, $\mathcal{W}^{(k)}$ es la matriz de pesos asociados a esa capa y $\mathcal{B}^{(k)}$ es el vector de bias de la misma.

- La red neuronal consistirá en una concatenación de n bloques sucesivos.

Sea $i \in \Gamma_n$, el i -ésimo bloque es la función realizada por las capas desde la $4i - 4$ hasta la $4i$, y lo notaremos por $\Phi[i]$. Por tanto, $\Phi[i] = \Phi^{(4i)} \circ \Phi^{(4i-1)} \circ \dots \circ \Phi^{(4i-4)}$, y $\Phi = \Phi[n] \circ \Phi[n-1] \circ \dots \circ \Phi[1]$. Cada bloque tendrá anchura $n + 2$ y profundidad 4.

- Cada bloque toma un vector de \mathbb{R}^{n+1} como entrada y produce un vector de \mathbb{R}^{n+1} como salida, excepto el primero, que toma un vector de \mathbb{R}^n (el input de la red global); y el último, que genera una salida en \mathbb{R} (la respuesta de la red global).

Los bloques quedan definidos como:

$$\begin{aligned}\Phi[1] &: \mathbb{R}^n \rightarrow \mathbb{R}^{n+1} \\ \Phi[i] &: \mathbb{R}^{n+1} \rightarrow \mathbb{R}^{n+1}, i \in \{2, \dots, n-1\} \\ \Phi[n] &: \mathbb{R}^{n+1} \rightarrow \mathbb{R}.\end{aligned}$$

- Cada capa de la red empleará n neuronas únicamente para recordar las n coordenadas del input inicial, y las dos restantes para realizar los cálculos pertinentes, que irán generando el resultado final en n etapas diferentes, cada una correspondiente con uno de los bloques descritos.
- La red se configurará a través de un parámetro δ . A medida que δ se acerca a 0, la función que realiza la red Φ aproximará con mayor precisión la función característica del cubo C , coincidiendo ambas en el límite.

Dado un $i \in \{2, \dots, n-1\}$, observamos la arquitectura del bloque $\Phi[i]$, que se refleja en el diagrama de la [Figura 3.3](#).

Fijamos una entrada de $\Phi[i]$ consistente en las n coordenadas de un punto $(x_1, x_2, \dots, x_n) = x \in \mathbb{R}^n$, donde $x_i \geq 0 \forall i \in \Gamma_n$; y un valor $z_{i-1} \in [0, 1]$. Al ser no negativos, dichos valores de entrada no se verán afectados cuando se les aplique una función de activación ReLU.

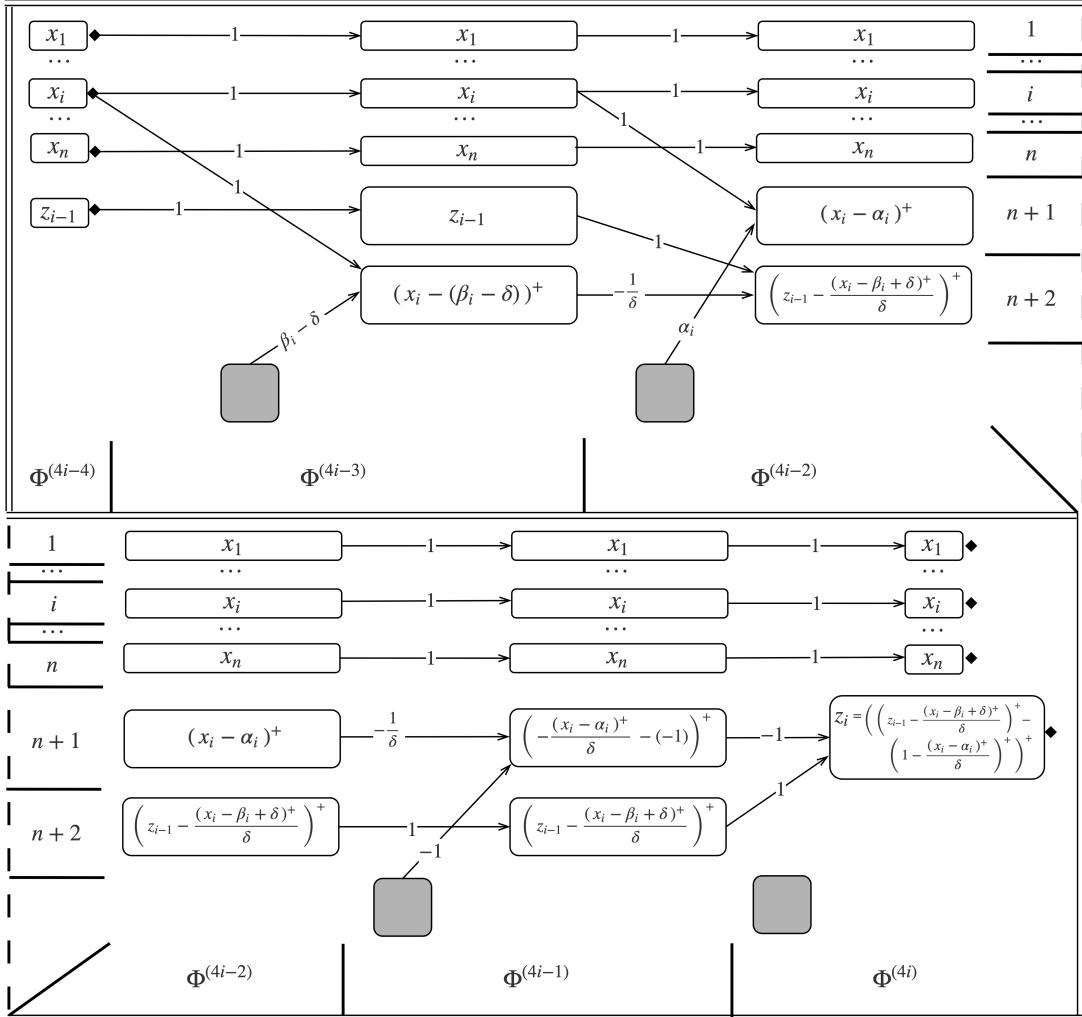


Figura 3.3: Esquema de la arquitectura neuronal del bloque $\Phi[i]$, con $i \in \{2, \dots, n-1\}$, tomando parámetros $\alpha_i, \beta_i \in \mathbb{R}_0^+$ con $\alpha_i \leq \beta_i$ y un $\delta \in]0, \min_{i \in \Gamma_n} \{\frac{\beta_i - \alpha_i}{2}\} [$. Se utiliza la representación vista en la Figura 2.1, utilizando en este caso formas rectangulares para poder incluir la expresión de las funciones de activación que realiza cada neurona. Se asume un input $x = (x_1, \dots, x_n) \in \mathbb{R}^n$ de coordenadas no negativas.

3 Capacidad de Cómputo de las Redes Neuronales

Por tanto, la salida de $\Phi[i]$ está conformada exactamente por las n coordenadas de x y un elemento z_i , cuyo valor se calcula a partir de z_{i-1} y x_i . En concreto, el valor z_i es la $(n+1)$ -ésima componente del output de $\Phi[i]$, es decir, la salida de la neurona $\Phi_{n+1}^{(4i)}$. Su expresión explícita viene dada por

$$z_i = \left(\left(z_{i-1} - \frac{(x_i - \beta_i + \delta)^+}{\delta} \right)^+ - \left(1 - \frac{(x_i - \alpha_i)^+}{\delta} \right)^+ \right)^+. \quad (3.2)$$

- Como $z_{i-1} \in [0, 1]$, el primer término de la resta es siempre menor o igual que 1, por lo que tras aplicar la función ReLU a la resta, es claro que $z_i \in [0, 1]$.
- Si $z_{i-1} = 0$, el primer término de la resta se anula (al ser una función ReLU aplicada a un valor negativo), por lo que se anula la resta completa tras aplicar ReLU, obteniendo $z_i = 0$.
- Si $z_{i-1} = 1$, la expresión (3.2) equivale a la siguiente:

$$z_i = \begin{cases} 0 & \text{si } x_i \leq \alpha_i \\ \frac{x_i - \alpha_i}{\delta} & \text{si } \alpha_i < x_i < \alpha_i + \delta \\ 1 & \text{si } \alpha_i + \delta \leq x_i \leq \beta_i - \delta \\ \frac{\beta_i - x_i}{\delta} & \text{si } \beta_i - \delta < x_i < \beta_i \\ 0 & \text{si } \beta_i \leq x_i. \end{cases}$$

Esta equivalencia se muestra gráficamente en la [Figura 3.4](#). Como vemos, el valor de z_i en función de x_i dibuja la forma de un trapecio cuyo lado mayor es $[\alpha_i, \beta_i]$, su lado menor es $[\alpha_i + \delta, \beta_i - \delta]$ y su altura es 1.

Por tanto, en este caso, $z_i = 1$ si y solo si $x_i \in [\alpha_i + \delta, \beta_i - \delta]$. Por el contrario, $z_i = 0$ si y solo si $x_i \notin [\alpha_i, \beta_i]$. En consecuencia, se tiene que cuando $\delta \rightarrow 0$, $z_i \rightarrow \chi_{[\alpha_i, \beta_i]}(x_i)$.

La arquitectura de $\Phi[n]$ será exactamente la misma que la descrita anteriormente, pero eliminando las n primeras neuronas de la capa $\Phi^{(4n)}$ para que el output de Φ sea únicamente el valor z_n calculado. El bloque $\Phi[1]$ también presentará la misma estructura, pero con algunas ligeras modificaciones:

- Puesto que vamos a utilizar funciones de activación ReLU en todas las neuronas y los valores negativos son filtrados a cero por éstas, vamos a trasladar el problema de forma que aseguremos que los inputs $x \in K$ se recuerdan correctamente a través de las capas de Φ , pudiéndose realizar los correspondientes cálculos de forma satisfactoria.

Dado un $x = (x_1, x_2, \dots, x_n) \in K$, tenemos que $x_i + N \geq 0$, por lo cual $x_i + N = (x_i + N)^+ \forall i \in \Gamma_n$. Notando $v_N = (N, N, \dots, N) \in \mathbb{R}^n$ es claro que $x \in C$ si, y solo si, $x + v_N \in C + v_N$. Pero además, como $x + v_N$ queda igual tras aplicar ReLU a todos sus componentes, se tiene que

$$x \in C \iff (x + v_N)^+ \in C + v_N \quad (3.3)$$

donde $(x + v_N)^+ = ((x_1 + N)^+, (x_2 + N)^+, \dots, (x_n + N)^+) \in \mathbb{R}^n$.

Por tanto, plantearemos la red Φ de forma que, internamente, aproxime $\chi_{C+v_N}((x + v_N)^+)$ en lugar de $\chi_C(x)$, ya que son equivalentes cuando $x \in K$.

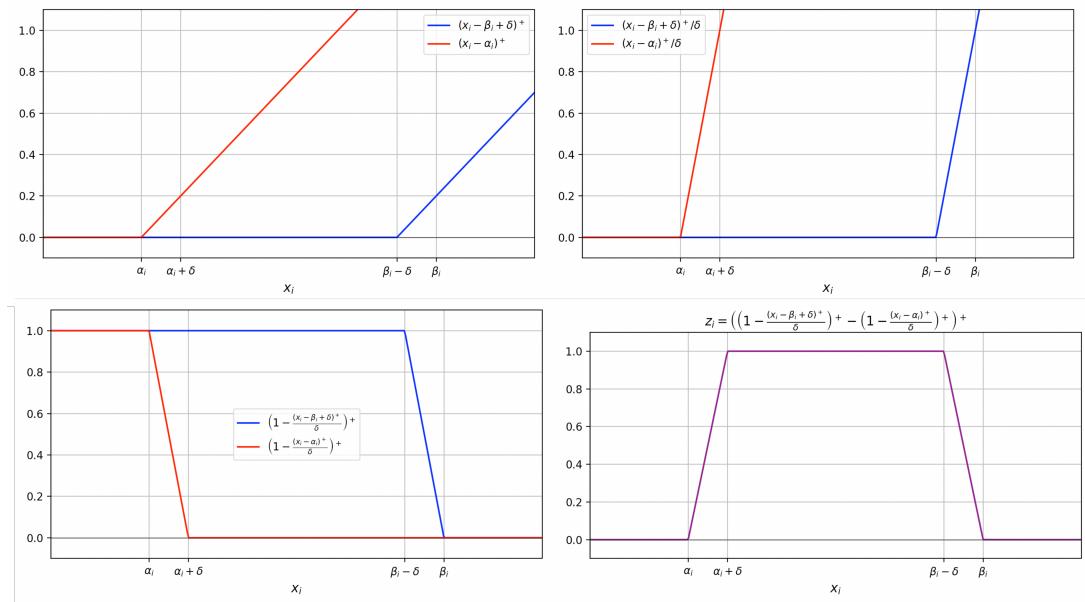


Figura 3.4: Descripción gráfica de la construcción del valor z_i en función de x_i , paso por paso, cuando $z_{i-1} = 1$. Obsérvese que cuanto más se aproxime a 0 el valor de δ , más cercana será la gráfica a la silueta de la función característica de $[\alpha_i, \beta_i]$.

Para ello, será necesario realizar una translación del input por v_N en la primera capa de Φ , que pertenece al bloque $\Phi[1]$. Sencillamente, dado el bias $\mathcal{B}^{(1)}$ de la primera capa de la red, tomamos

$$\mathcal{B}_i^{(1)} = -N \quad \forall i \in \Gamma_n.$$

De esta forma, para cada $i \in \Gamma_n$, se tiene que $\Phi_i^{(1)}(x) = (x_i + N)^+ \forall x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$, y será este valor (no negativo) el que dejarán pasar las neuronas ReLU a través de las capas de Φ . Los bloques sucesivos reciben las coordenadas de x ya trasladadas y filtradas. Otro pequeño detalle a tener en cuenta para mantener la consistencia, es que debemos tomar el bias

$$\mathcal{B}_{n+2}^{(1)} = \beta_1 - \delta - N \tag{3.4}$$

para que la neurona $\Phi_{n+2}^{(1)}$ calcule exactamente el valor $((x_1 + N) - (\delta - \beta_1))^+$.

Si $x \notin K$, puede ocurrir que alguna coordenada de $x + v$ se haya filtrado a cero siendo negativa, por lo que se pierde información. Si $x \in K$, las n primeras neuronas de cada capa de Φ recordarán exactamente el vector $x + v$, manteniéndolo intacto hasta la última capa.

- El bloque $\Phi[1]$ va a recibir únicamente n valores en su primera capa $\Phi^{(0)}$, que se corresponden con las n coordenadas de un punto $x \in \mathbb{R}^n$. El valor z_0 que se utiliza en la expresión (3.2) para calcular z_1 no se recibirá como input, sino que se establecerá por defecto en 1 haciendo uso del bias. En concreto, se añade $\mathcal{B}_{n+2}^{(2)} = -1$, dejando la neurona $\Phi_n^{(1)}$ inutilizada.

Dado un cubo $C \subset K$, consideramos para cada $i \in \Gamma_n$ los extremos de la i -ésima proyección

de C . Estos son $\alpha'_i = \inf \pi_i(C)$, $\beta'_i = \sup \pi_i(C)$, que cumplen que $\alpha'_i, \beta'_i \in K$ y $\alpha'_i \leq \beta'_i$.

Ahora, definimos $\alpha_i = \alpha'_i + N$, $\beta_i = \beta'_i + N \forall i \in \Gamma_n$. Estos valores son, exactamente, los extremos de las proyecciones del cubo $C + v_N$, que por ello cumplen que $\alpha_i, \beta_i \in \mathbb{R}_0^+$, $\alpha_i \leq \beta_i$. Ahora, dado un $\delta \in]0, \min_{i \in \Gamma_n} \{\frac{\beta_i - \alpha_i}{2}\} [$, definimos el cubo

$$C_\delta = \prod_{i=1}^n [\alpha'_i + \delta, \beta'_i - \delta].$$

Claramente $\text{int}(C) = \bigcup_{k=1}^{\infty} C_{(\frac{1}{k})}$ con $C_{(\frac{1}{k})} \subset C_{(\frac{1}{k+1})}$, y por continuidad creciente de μ , se tiene que $\lim_{k \rightarrow \infty} \mu(C_{(\frac{1}{k})}) = \mu(\text{int}(C)) = \mu(C)$. En concreto, dado un $\epsilon > 0$,

$$\exists k \in \mathbb{N} \text{ tal que } \delta = \frac{1}{k} \implies \mu(C) - \mu(C_\delta) < \epsilon. \quad (3.5)$$

Para cada $i \in \Gamma_n$, definimos el bloque $\Phi[i]$ utilizando los parámetros α_i y β_i indicados, junto con el valor δ de (3.5). En estas condiciones, definimos la red neuronal ReLU totalmente conectada $\Phi = \Phi[n] \circ \Phi[n-1] \circ \dots \circ \Phi[1]$, la cual recibe una entrada $(x_1, x_2, \dots, x_n) = x \in \mathbb{R}^n$ y devuelve un único valor en $\Phi(x) \in \mathbb{R}$. Estudiemos dicho valor:

- Como z_0 se establece en 1 por defecto, se sabe que $\Phi[1]$ calcula un valor $z_1 \in [0, 1]$, el cual será la $(n+1)$ -ésima componente del input de $\Phi[2]$. Entonces, $z_2 \in [0, 1]$. Repitiendo este proceso, se llega a que $z_n = \Phi(x) \in [0, 1]$. Esto es, la función Φ sólo toma valores en $[0, 1]$.
- Si $x \in C_\delta \subset K$, se tiene $(x + v_N)^+ \in C_\delta + v_N$ por (3.3).

Entonces $(x_1 + N)^+ \in [\alpha_1 + \delta, \beta_1 - \delta]$, por lo que $\Phi[1]$ calcula el valor $z_1 = 1$, y este será proporcionado a $\Phi[2]$ como $(n+1)$ -ésima componente de su input. Igualmente, $(x_2 + N)^+ \in [\alpha_2 + \delta, \beta_2 - \delta]$, tenemos que $z_2 = 1$. Inductivamente, se comprueba que $z_n = \Phi(x) = 1$.

- Si $x \notin C$, hay dos casos:

- Si $x + v_N = (x + v_N)^+$, se tiene que $(x + v_N)^+ \notin C + v_N$. Entonces existe $i \in \Gamma_n$ tal que $(x_i + N)^+ \notin [\alpha_i, \beta_i]$. Esto implica que $z_i = 0$, y al darse como entrada de $\Phi[i+1]$, se calcula $z_{i+1} = 0$. Aplicando inducción, queda que $z_n = \Phi(x) = 0$.
- Si $x + v_N \neq (x + v_N)^+$, es porque la función ReLU trunca a cero alguna de las coordenadas de $x + v_N$. Es decir, $(x_i + N)^+ = 0$ para algun $i \in \Gamma_n$. En tal caso, el bloque $\Phi[i]$ calcula $z_i = 0$, ya que usando la expresión (3.2), $z_i = (z_{i-1} - 1)^+ = 0$. Se puede aplicar inducción como antes para ver que, efectivamente, $z_n = \Phi(x) = 0$. Nótese que este caso solo sucede si $x \notin K$.

La función $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}$ es medible puesto que su salida se calcula aplicando sumas, restas y la función continua ReLU de manera recursiva sobre las coordenadas del input $x \in \mathbb{R}^n$. Teniendo en cuenta el desarrollo anterior, se puede ver que $\Phi(x) = \chi_C(x)$ si $x \in C_\delta \cup C^c$. Obviamente, cuando $\delta \rightarrow 0$, se tiene que $C_\delta \rightarrow C$, por lo que es cierto que $\lim_{\delta \rightarrow 0} \Phi(x) = \chi_C(x) \forall x \in \mathbb{R}^n$. Además, si $x \in C \setminus C_\delta$, sabemos que $\chi_C(x) = 1$ y $\Phi(x) \in [0, 1]$ por lo cual $|\chi_C - \Phi| = \chi_C - \Phi \in \mathcal{L}^+$, y por las propiedades de la integral en \mathcal{L}^+ ,

$$\mu(C_\delta) = \int_{\mathbb{R}^n} \chi_{C_\delta} \leq \int_{\mathbb{R}^n} \Phi \leq \int_{\mathbb{R}^n} \chi_C = \mu(C) < \infty,$$

lo que nos dice que $\Phi \in \mathcal{L}^1$. Usando todo ello y (3.5), se verifica la desigualdad deseada

$$\|\chi_C - \Phi\|_{\mathcal{L}^1} = \int_{\mathbb{R}^n} |\chi_C - \Phi| = \int_{\mathbb{R}^n} (\chi_C - \Phi) = \int_{\mathbb{R}^n} \chi_C - \int_{\mathbb{R}^n} \Phi \leq \mu(C) - \mu(C_\delta) < \epsilon,$$

concluyendo así la demostración. \square

3.2.2. Aproximación de una combinación lineal de funciones características de cubos

Lema 3.2. *Sea una función $S := \sum_{j=1}^m \delta_j \cdot \chi_{C_j} \in \mathcal{L}^1$, donde $m \in \mathbb{N}$ y para cada $j \in \Gamma_m$, $\delta_j \in \mathbb{R}$ y C_j es un cubo de \mathbb{R}^n contenido en $K = [-N, N]^n$ para cierto $N > 0$. Entonces, existe una red neuronal ReLU totalmente conectada de anchura $w = n + 4$ y profundidad $d = 4nm + 2$ que realiza una función $g \in \mathcal{L}^1$ de forma que, dado un $\epsilon > 0$,*

$$\|S - g\|_{\mathcal{L}^1} < \epsilon.$$

Demostración. En primer lugar, para cada $j \in \Gamma_m$, consideramos una red neuronal ReLU totalmente conectada Φ_{C_j} como la que se construyó en el Lema 3.1, verificando que

$$\|\chi_{C_j} - \Phi_{C_j}\|_{\mathcal{L}^1} < \frac{\epsilon}{m \cdot |\delta_j|}.$$

El objetivo será concatenar estas redes de alguna manera para construir una red g cuyo input sea un $x \in \mathbb{R}^n$ y su output sea exactamente $g(x) = \sum_{j=1}^m \delta_j \cdot \Phi_{C_j}(x)$. Si conseguimos hacerlo, como $\Phi_{C_j} \in \mathcal{L}^1 \forall j \in \Gamma_m$, tendremos $g \in \mathcal{L}^1$ por ser \mathcal{L}^1 un espacio vectorial, luego g cumplirá que:

$$\begin{aligned} \|S - g\|_{\mathcal{L}^1} &= \left\| \sum_{j=1}^m \delta_j \cdot \chi_{C_j} - \sum_{j=1}^m \delta_j \cdot \Phi_{C_j} \right\|_{\mathcal{L}^1} = \left\| \sum_{j=1}^m \delta_j \cdot (\chi_{C_j} - \Phi_{C_j}) \right\|_{\mathcal{L}^1} \\ &\leq \sum_{j=1}^m |\delta_j| \cdot \|\chi_{C_j} - \Phi_{C_j}\|_{\mathcal{L}^1} < \sum_{j=1}^m |\delta_j| \cdot \frac{\epsilon}{m \cdot |\delta_j|} = \sum_{j=1}^m \frac{\epsilon}{m} = \epsilon. \end{aligned} \quad (3.6)$$

Veamos los detalles técnicos para conseguir dicha red neuronal g , que tendrá anchura $w = n + 4$ y profundidad $d = 4nm + 2$. Dado $k \in \Gamma_{4nm+2}$, notamos por $g^{(k)}$ a su k -ésima capa, $\mathcal{W}^{(k)}$ a la matriz de pesos de dicha capa, y por $\mathcal{B}^{(k)}$ al vector de bias de la misma.

- En primer lugar, añadiremos n neuronas ReLU en la última capa de Φ_{C_j} , para cada $j \in \Gamma_m$. La neurona que calculaba su output se situará como la $(n + 1)$ -ésima de dicha capa. Además, por conveniencia en la notación, se asumirá la existencia de una $(n + 2)$ -ésima neurona ReLU en esta misma capa, cuyos pesos asociados serán nulos. En estas condiciones, ya podemos describir la red $g = \Phi_{C_m} \circ \Phi_{C_{m-1}} \circ \dots \circ \Phi_{C_1}$ como una función de \mathbb{R}^n en \mathbb{R}^{n+2} . Una vez que tenemos esta estructura, añadimos los pesos

$$\mathcal{W}_{i,i}^{(4nj)} = 1 \quad \forall i \in \Gamma_n, \forall j \in \Gamma_m.$$

Así mismo, ponemos algunos bias de la primera capa de Φ_{C_j} a cero $\forall j \in \Gamma_m \setminus \{1\}$, puesto que sólo realizaremos la translación y filtrado de las coordenadas del input en la

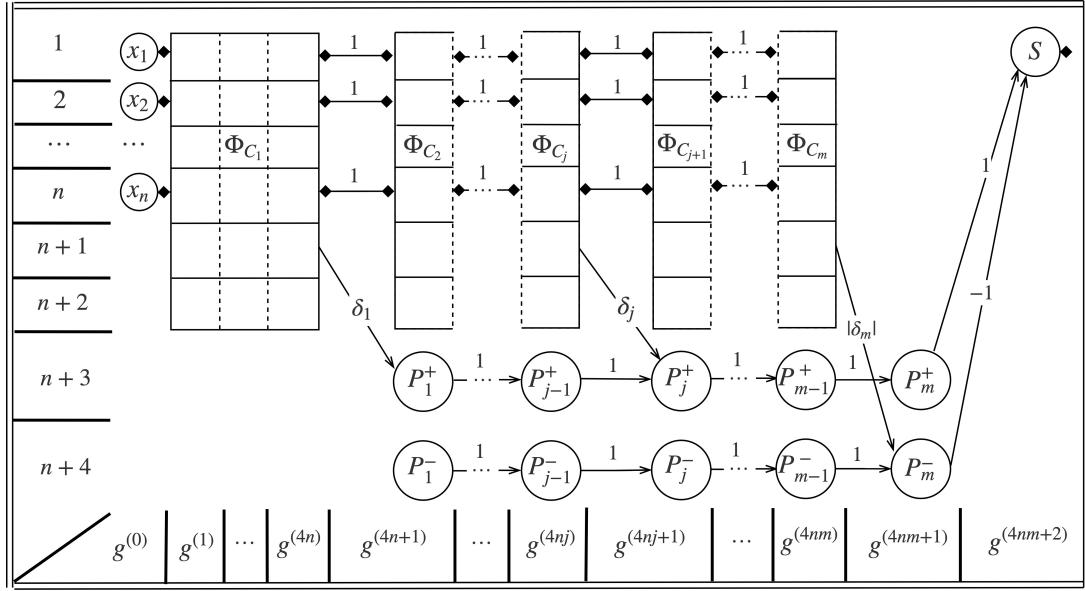


Figura 3.5: Esquema de la configuración de la red neuronal g , de anchura $w = n + 4$ y profundidad $d = 4nm + 2$. Las redes de la forma Φ_{C_j} para cada $j \in \Gamma_m$ se simbolizan mediante rectángulos. Las líneas discontinuas dentro de los mismos indican la separación entre capas internas de dichas redes. No se representa el bias puesto que no afecta a ninguna de las nuevas neuronas incluidas en la arquitectura. Dado $j \in \Gamma_m$, P_j^+ denota la suma acumulada de términos positivos de S tras realizar la red Φ_{C_j} , y P_j^- la suma acumulada de términos negativos. Hemos supuesto $\delta_1 \geq 0$, $\delta_j \geq 0$ y $\delta_m < 0$.

primera capa de g . En concreto,

$$\mathcal{B}_i^{(4nj+1)} = 0 \quad \forall i \in \Gamma_n, \forall j \in \Gamma_{m-1}.$$

Además, tendremos que sumar N al bias $\mathcal{B}_{n+2}^{(4nj+1)}$ para cada $j \in \Gamma_{m-1}$, para revertir el efecto descrito en (3.4).

Con estas modificaciones, ya tenemos una red donde la neurona $g_{n+1}^{(4nj)}$, $j \in \Gamma_m$, está calculando $\Phi_{C_j}(x)$ para un input $x \in \mathbb{R}^n$ de g .

- Añadimos dos neuronas ReLU $g_{n+3}^{(k)}$ y $g_{n+4}^{(k)}$ en la capa k , $\forall k \in \{4n + 1, \dots, 4nm + 1\}$. Estas neuronas servirán para recordar e ir calculando la parte positiva y negativa de la suma, respectivamente. Como se muestra en la Figura 3.5, añadimos pesos $\mathcal{W}_{n+3, n+3}^{(k)} = 1$ y $\mathcal{W}_{n+4, n+4}^{(k)} = 1$ $\forall k \in \{4n + 1, \dots, 4n \cdot m + 1\}$. Estos pesos simplemente sirven para ir trasladando las cantidades acumuladas calculadas entre capas hasta llegar al final de la red. Ahora, fijando un input $x \in \mathbb{R}^n$ de la red g y $j \in \Gamma_m$, distinguimos dos casos:
 - Si $\delta_j \geq 0$, añadimos $\mathcal{W}_{n+1, n+3}^{(4nj+1)} = \delta_j$ y dejamos $\mathcal{W}_{n+1, n+4}^{(4nj+1)} = 0$. En este caso, el valor $\delta_j \cdot \Phi_{C_j}(x)$ llega a la $(n+3)$ -ésima neurona de la capa $g^{(4nj+1)}$; y un cero llega a la $(n+4)$ -ésima neurona de esta misma capa.

- De modo similar, si $\delta_j < 0$, incluimos $\mathcal{W}_{n+1,n+4}^{(4nj+1)} = |\delta_j|$ y ponemos $\mathcal{W}_{n+1,n+3}^{(4nj+1)} = 0$, llegando pues $|\delta_j| \cdot \Phi_{C_j}(x)$ a $g_{n+4}^{(4nj+1)}$ y cero a $g_{n+3}^{(4nj+1)}$.

En cualquier caso, esta cantidad se suma con la que viene de las anteriores capas y ha sido trasladada mediante los pesos unitarios definidos (excepto en $g^{(4n+1)}$, que es la primera capa en la que se definen estas nuevas neuronas, y por tanto solo recibe el valor de salida de $g_{n+1}^{(4n)}$). Evidentemente, esta suma es no negativa puesto que $\Phi_{C_j}(x) \in [0, 1]$, luego las funciones ReLU no la truncan a través de las capas.

Es sencillo comprobar que, siguiendo esta lógica, al llegar a la capa $4nm + 1$, su neurona $n + 3$ calcula la parte positiva de $\sum_{j=1}^m \delta_j \cdot \Phi_{C_j}$ y su neurona $n + 4$, su parte negativa. Nuevamente, para conservar la coherencia, debemos suponer la presencia de neuronas ReLU $g_1^{(4nm+1)}, \dots, g_{n+1}^{(4nm+1)}, g_{n+2}^{(4nm+1)}$, cuyos cálculos no afectarán al output de la red.

- Para finalizar y dar la salida deseada, añadimos una última capa $g^{(4nm+2)}$ que consta de una única neurona. Esta neurona recibe el valor de salida de $g_{n+3}^{(4nm+1)}$ multiplicado por el peso 1 y el de $g_{n+4}^{(4nm+1)}$ multiplicado por el peso -1 . Esta será la única neurona de toda la arquitectura que no utiliza función de activación ReLU: simplemente devuelve el resultado de la suma, que puede ser positivo o negativo.

Entonces, hemos descrito una red neuronal ReLU totalmente conectada que realiza una función $g : \mathbb{R}^n \rightarrow \mathbb{R}$ de forma que exactamente $g(x) = \sum_{j=1}^m \delta_j \cdot \Phi_{C_j}(x) \forall x \in \mathbb{R}^n$. Aplicando (3.6), hemos terminado. \square

3.2.3. Teorema de Aproximación Universal para redes ReLU limitadas por anchura

Teorema 3.3. *Sea $\epsilon > 0$ y $f \in \mathcal{L}^1$. Existe una red neuronal ReLU totalmente conectada con anchura $w \leq n + 4$, de forma que la función $g : \mathbb{R}^n \rightarrow \mathbb{R}$ que realiza cumple que*

$$\|f - g\|_{\mathcal{L}^1} < \epsilon.$$

Demostración. En primer lugar, sabiendo que $f \in \mathcal{L}^1$, podemos construir una familia finita de cubos $\{C_j\}_{j=1}^m$, $m \in \mathbb{N}$, y tomar escalares $\delta_1, \delta_2, \dots, \delta_m \in \mathbb{R}$ de forma que la función

$$S := \sum_{j=1}^m \delta_j \cdot \chi_{C_j} \in \mathcal{L}^1 \text{ verifica que } \|f - S\|_{\mathcal{L}^1} < \frac{\epsilon}{2}, \quad (3.7)$$

en virtud del Corolario 1.1.1. Además, es posible seleccionar $N > 0$ tal que $\bigcup_{j=1}^m C_j \subset [-N, N]^n$, por ser $\{C_j\}_{j=1}^m$ una familia finita de cubos (acotados).

La función $S \in \mathcal{L}^1$ de (3.7) es, por tanto, una combinación lineal finita de funciones características sobre cubos de \mathbb{R}^n contenidos en $[-N, N]^n$, por lo que aplicando el Lema 3.2, encontramos una función $g \in \mathcal{L}^1$ realizada por una red neuronal ReLU totalmente conectada con anchura $w = n + 4$ de forma que

$$\|S - g\|_{\mathcal{L}^1} < \frac{\epsilon}{2}. \quad (3.8)$$

Usando (3.7) y (3.8) junto con la desigualdad triangular de $\|\cdot\|_{\mathcal{L}^1}$, concluimos que

$$\|f - g\|_{\mathcal{L}^1} \leq \|f - S\|_{\mathcal{L}^1} + \|S - g\|_{\mathcal{L}^1} < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon.$$

□

Este resultado nos dice que cualquier función integrable de Lebesgue puede ser aproximada, hasta cierto grado de precisión arbitrario (con respecto de la distancia en \mathcal{L}^1), mediante una red neuronal totalmente conectada con anchura acotada linealmente y funciones de activación ReLU. Este tipo de aproximación también es coherente en términos prácticos, pues conforme el error tiende a cero, el número de puntos donde la función integrable presentará diferencias significativas con la función que realiza la red tendrá cada vez medida más pequeña.

3.2.4. Pérdida de expresividad de las redes ReLU limitadas por anchura

En el artículo [37], encontramos otros dos importantes resultados que se encuadran en el contexto de análisis de expresividad de redes ReLU acotadas en anchura, y que se presentan a continuación.

Teorema 3.4. *Sea $f \in \mathcal{L}^1$ tal que $\mu(\{x : f(x) \neq 0\}) > 0$, y cualquier función $g : \mathbb{R}^n \rightarrow \mathbb{R}$ realizada por una red neuronal ReLU totalmente conectada con anchura $w \leq n$, se cumple que*

$$\int_{\mathbb{R}^n} |f - g| = +\infty \quad o \quad \int_{\mathbb{R}^n} |f - g| = \|f\|_{\mathcal{L}^1}$$

El Teorema 3.4 nos dice que ninguna red ReLU totalmente conectada con anchura menor o igual que n , siendo n el tamaño de la entrada, es capaz de aproximar una función cualquiera de \mathcal{L}^1 por debajo de cierto límite, que viene dado por seminorma de dicha función en \mathcal{L}^1 .

Podríamos pensar que quizás esto ocurre por considerar funciones integrables definidas en todo el espacio euclidiano \mathbb{R}^n , pero también se puede probar un resultado similar para conjuntos acotados de \mathbb{R}^n :

Teorema 3.5. *Sea $f : [-1, 1]^n \rightarrow \mathbb{R}$ continua y no constante en ninguna dirección, existe un $\epsilon^* > 0$ universal tal que para cualquier función $g : [-1, 1]^n \rightarrow \mathbb{R}$ representada por una red ReLU totalmente conectada con anchura $w \leq n - 1$, se cumple que*

$$\int_{\mathbb{R}^n} |f - g| \geq \epsilon^*$$

De nuevo, nos encontramos con un límite en la aproximación cuando rebajamos suficientemente la anchura de la arquitectura con respecto del tamaño del input. Por lo tanto, se evidencia que una anchura escasa en una arquitectura neuronal representa una barrera significativa para su potencial.

Estos resultados contrastan con los examinados en la Sección 3.1, donde demostrábamos que una sola capa (profundidad $d = 2$), era suficiente para aproximar una clase muy extensa de funciones: la de las funciones continuas en compactos, independientemente de la dimensionalidad del compacto de definición.

De manera preliminar, y obviando las diferencias entre los resultados, pudiera parecer que esto indica que la profundidad de la red confiere al modelo neuronal de un mayor potencial que el que puede proporcionar su anchura. Dicho de otro modo, la anexión de más capas a una red puede ser más efectiva que la adición de más neuronas en las capas

presentes. La reciente inclinación hacia el estudio de redes profundas (Deep Learning) [30] y su sorprendente rendimiento en problemas reales refuerza este argumento. En esta línea, surge la siguiente pregunta:

"Si una red neuronal ReLU totalmente conectada reduce su anchura, ¿cuánto ha de aumentar, como mínimo, la profundidad de la misma para compensarla sin pérdida de capacidad expresiva?"

El problema dual es el análisis de la **eficiencia de las Redes Profundas**, de la cual se conocen resultados muy reveladores. En concreto, existen teoremas que nos muestran que el hecho de suprimir capas en una red neuronal ReLU fuerza, al menos, a un aumento **exponencial** en su anchura [50, 15] para poder replicar su capacidad de representación.

La posible existencia de un resultado análogo en la otra dirección se engloba en el contexto de la **eficiencia de las Redes Anchas**. Si se probase que una reducción de anchura se traduce, como mínimo, en un aumento exponencial de la profundidad de una arquitectura para evitar una pérdida en su potencial de aproximación de funciones, podríamos argumentar que el peso de la anchura y la profundidad es equivalente en términos de expresividad (al menos en el terreno teórico).

Desafortunadamente, en [37] no se provee una justificación teórica fehaciente que avale esta tesis. Simplemente se prueba la existencia de un límite inferior de orden **polinómico** en el incremento de la profundidad, por debajo del cual se pierde potencial, y se deja abierta la posible existencia de un resultado análogo como el descrito; o bien la de un límite superior polinómico por debajo del cual siempre se puede aproximar la función realizada por la red original. En este último caso, se podría razonar que el rol de la anchura es secundario con respecto de la profundidad, como parecen indicar las múltiples observaciones empíricas. De hecho, en el artículo se plantea el siguiente experimento:

1. Se definen una serie de arquitecturas neuronales de anchura $w = 2k^2$, con $k \in \{3, 4, 5\}$, y profundidad $d = 3$, a las que llamaremos **redes anchas**. Además, todas ellas se implementan paralelamente para entradas unidimensionales ($n = 1$) o bidimensionales ($n = 2$), produciendo una única salida en todos los casos.
2. Se establecen pesos y biases aleatorios para dichas arquitecturas. Los pesos se obtienen de una distribución normal estándar, y los biases de una distribución uniforme en $[-1, 1]$. Cada una de las redes anchas con sus respectivos parámetros define una **función objetivo**. Se toma un número grande de muestras distribuidas uniformemente en $[-1, 1]^n$, las cuales se evalúan mediante la función objetivo definida por cada red, dando lugar a pares etiquetados que constituyen conjuntos de entrenamiento.
3. Para cada red ancha definida por $k \in \{3, 4, 5\}$ y $n \in \{1, 2\}$, se plantea una nueva arquitectura de anchura $w = 3k^{\frac{3}{2}}$ y profundidad $d = k + 2$, a la que denominaremos **red estrecha**. Esto es, la profundidad de la red estrecha aumenta de manera **polinómica** en relación con la disminución de su anchura con respecto de la red ancha.
4. Se entrena cada red estrecha usando un enfoque clásico de Gradiente Descenso, con pesos iniciales aleatorios, y utilizando el conjunto de entrenamiento generado mediante la red ancha correspondiente. De esta forma, la red estrecha se entrena con el fin de modelar la función realizada por la red ancha. Tras el entrenamiento, se obtienen medidas de la bondad de la aproximación de la función objetivo mediante el Error Cuadrático Medio (MSE).

5. Para mayor consistencia, se repite el procedimiento un total 50 veces para cada caso, tomando diferentes parametrizaciones iniciales de la red ancha concreta. Al final, se toma una **media del MSE** calculado para todas las aproximaciones de la red ancha con las distintas parametrizaciones, así como el **peor MSE** de entre todas las aproximaciones obtenidas mediante la red estrecha correspondiente.
6. Se comprueba que el aumento de orden polinómico en la profundidad en la red estrecha es suficiente para emular la función objetivo realizada por la red ancha con gran precisión. Esto sugiere que, efectivamente, una compensación polinómica de la profundidad es suficiente para contrarrestar la pérdida de capacidad expresiva derivada de la rebaja de neuronas por capa.

A pesar de estas evidencias, resulta interesante acometer el problema abierto que propone el trabajo con rigor matemático. Algunos investigadores ya lo han abordado. Concretamente, en 2022 se publicó el documento [60], de nombre "*Width is Less Important than Depth in ReLU Neural Networks*", que precisamente propone con éxito una solución del mismo. En concreto, se alcanza el siguiente teorema:

Teorema 3.6. *Sea $\mathcal{A}_0 : \mathbb{R}^n \rightarrow \mathbb{R}$ una red neuronal ReLU totalmente conectada con anchura $w \in \mathbb{N}$, profundidad $d \in \mathbb{N}$, y sea \mathcal{X} alguna distribución de entrada con una función de densidad acotada superiormente sobre un dominio acotado en \mathbb{R}^n . Entonces, para cada $\epsilon, \delta > 0$ existe una red neuronal $\mathcal{A} : \mathbb{R}^n \rightarrow \mathbb{R}$ con anchura $O(n)$ y $\tilde{O}(w^2d^2)$ parámetros de tal forma que, con probabilidad al menos $1 - \delta$ sobre $\mathbf{x} \sim \mathcal{X}$, se cumple que*

$$|\mathcal{A}_0(\mathbf{x}) - \mathcal{A}(\mathbf{x})| \leq \epsilon$$

Observación 3.2.1. La notación **big-O** (O) describe un límite superior en la tasa de crecimiento de una función, ignorando factores constantes. Formalmente,

$$f(n) = O(g(n)) \quad \text{si} \quad |f(n)| \leq C \cdot |g(n)|$$

para algún $C \in \mathbb{R}$ y $n \geq n_0 \in \mathbb{N}$. La notación **tilde-O** (\tilde{O}) es similar, pero oculta factores logarítmicos. Si $f(n) = \tilde{O}(g(n))$, entonces $f(n) = O(g(n) \cdot \log^k n)$ para algún $k \in \mathbb{N}$. En el teorema anterior, $\tilde{O}(w^2d^2)$ indica que el número de parámetros es proporcional a w^2d^2 con factores logarítmicos omitidos.

El **Teorema 3.6** tiene varias particularidades. Entre ellas, se introduce una distribución de probabilidad en la ecuación, y no se usa la distancia en \mathcal{L}^1 . Además, la demostración del mismo es muy avanzada. Sin embargo, nos quedaremos con la idea general que sugiere este resultado en comparación con los vistos anteriormente: por lo general, un aumento en la profundidad de una red le dota de mayores capacidades que una expansión en anchura de la misma magnitud, pudiendo inferir así que el papel de la anchura es menos significativo que el de la profundidad en arquitecturas neuronales ReLU.

4 Construcción Explícita de Redes Neuronales

En el [Capítulo 3](#), se presentan una serie de resultados teóricos que justifican la potencia expresiva de las redes neuronales. En concreto, hacemos hincapié en las redes neuronales ReLU totalmente conectadas y con anchura limitada, proporcionando instrucciones concretas para construir, de manera directa y explícita, arquitecturas de este tipo para aproximar funciones integrables con un factor de aproximación arbitrario.

Sin embargo, cuando intentamos trasladar este argumento al terreno computacional, que es donde el concepto de red neuronal cobra sentido, surgen numerosos impedimentos y consideraciones. El objetivo de este capítulo es discutir hasta qué punto y en qué condiciones se pueden replicar las configuraciones previamente desarrolladas en la práctica, y realizar un estudio de las posibles utilidades y/o aplicaciones de este enfoque en problemas de la vida real.

Para ello, analizaremos un código implementado específicamente para dicho propósito, y que se encuentra completo y con mayor detalle en [\[44\]](#). En este, se utiliza el lenguaje *Python* y librerías de amplia utilización en este contexto, como *Numpy*, *Pandas* o *Tensorflow*.

4.1. Funciones de interés

En la [Sección 3.1](#) y la [Sección 3.2](#), se discutían sendas construcciones neuronales que aproximan funciones, en distintos sentidos. Sin embargo, aunque el enfoque sea diferente, la base de ambos resultados es la misma: la capacidad de las funciones simples de aproximar funciones muy variadas. No obstante, a pesar de que la teoría nos dice que estas funciones simples existen, calcularlas puede resultar muy costoso en términos computacionales.

Para funciones continuas definidas en conjuntos compactos, la [Proposición 1.7.1](#) nos indica una forma concreta de calcular una partición del compacto de definición, de manera que podamos construir la función simple que la aproxima, la cual a su vez simularíamos mediante una red neuronal. El problema es que la partición descrita depende directamente de la forma de la propia función. Como las funciones continuas pueden variar arbitrariamente en entornos muy reducidos (véase la [Figura 4.1](#)), no podemos asegurar que una partición concreta, por fina que sea, vaya a dar lugar a la construcción de un buen estimador general para todas las funciones de este tipo.

Además, el aumento de tamaño de la partición se traduce, según lo visto, en un incremento de la anchura/profundidad de la red asociada, que en ocasiones podría tornarse inmanejable en la práctica. Por tanto, aunque uno de estos dos parámetros esté controlado, el otro podría resultar ser arbitrariamente grande.

Por ende, conviene escoger una clase de funciones con la que podamos trabajar con garantías en este sentido. Esta clase es la de las **funciones lipschitzianas** definidas en un conjunto acotado (y medible) de \mathbb{R}^n , que se presentan en la [Subsección 1.7.4](#). La variación de estas funciones está acotada en cualquier entorno, por lo que podemos establecer una partición del conjunto de definición conformada por un número acotado de conjuntos, según una precisión prefijada, de forma que dicha partición sea suficiente para aproximar cualquier

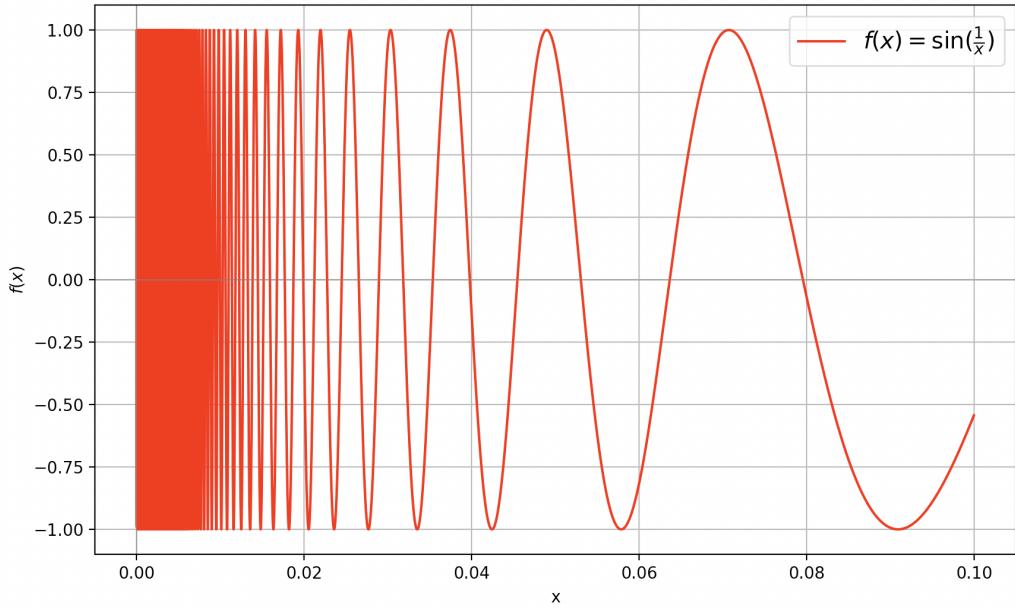


Figura 4.1: Gráfica de la función $f :]0, \frac{1}{10}] \rightarrow \mathbb{R}$ con $f(x) = \sin(\frac{1}{x}) \forall x \in]0, \frac{1}{10}]$. Esta función es continua en su dominio, pero la oscilación de la misma no está acotada conforme nos acercamos a $x = 0$.

función de la clase (con cierta constante de Lipschitz $L > 0$) haciendo uso de una función simple. Además, el criterio de aproximación puede ser el de la distancia puntual uniforme o el de la distancia definida en el espacio de las funciones integrables. Toda esta información se condensa en la [Proposición 1.7.2](#). Nosotros nos vamos a centrar en el segundo criterio, que es el que hemos estudiado en mayor profundidad.

En resumen, usando las ideas discutidas en el [Capítulo 3](#), implementaremos arquitecturas que sean capaces de proveer resultados muy similares a los ofrecidos por una amplia gama de funciones, pudiendo controlar adecuadamente el tamaño de estas arquitecturas de forma que este sea razonable.

4.2. Determinación de una partición

De ahora en adelante, trabajaremos en el contexto de las funciones lipschitzianas de la forma $f : I \rightarrow \mathbb{R}$, con $I = [a, b] \subset \mathbb{R}$, $a, b \in \mathbb{R}$, $a < b$, y constante de Lipschitz $L > 0$. Tomamos funciones de variable única por simplicidad, pero todos los argumentos que expondremos se pueden extender de manera relativamente sencilla al caso n -dimensional, $n > 1$.

Típicamente, en problemas relacionados con redes neuronales, disponemos de un número limitado de datos etiquetados, los cuales repartimos de forma equitativa entre un conjunto de entrenamiento, uno de validación y otro de evaluación. En términos de funciones, esto significa que contamos con una familia de pares $\{(x_i, f(x_i))\}_{i=1}^k$, $k \in \mathbb{N}$. Es evidente que, si el número de datos es insuficiente, no podremos garantizar una aproximación arbitraria de

la función concreta a la que corresponden estos datos. Lo que sí que podemos determinar, en estas condiciones, es una cota máxima de error (medido en nuestro caso mediante la distancia en $\mathcal{L}^1(I)$) de la función original, de cuya gráfica solo conocemos k puntos, con respecto de una función simple de $k - 1$ términos que confeccionamos específicamente.

4.2.1. Cálculo del error

Notemos en primer lugar que si tomamos $\alpha, \beta \in \mathbb{R}$, $\alpha < \beta$, de forma que conocemos sus valores $f(\alpha)$ y $f(\beta)$ por una función f lipschitziana, entonces la distancia $\left\| f - \frac{f(\alpha) + f(\beta)}{2} \right\|_{\mathcal{L}^1([\alpha, \beta])}$ está acotada, independientemente de cómo se comporte f en $[\alpha, \beta]$. Este hecho se deriva de la condición de Lipschitz (1.22), ya que

$$\begin{aligned} |f(x) - f(\alpha)| &\leq L \cdot |x - \alpha| \quad \forall x \in [\alpha, \beta] \iff \\ \iff f(x) &\leq f(\alpha) + L \cdot (x - \alpha) \wedge f(x) \geq f(\alpha) - L \cdot (x - \alpha) \quad \forall x \in [\alpha, \beta], \end{aligned}$$

y similarmente,

$$\begin{aligned} |f(x) - f(\beta)| &\leq L \cdot |\beta - x| \quad \forall x \in [\alpha, \beta] \iff \\ \iff f(x) &\leq f(\beta) - L \cdot (\beta - x) \wedge f(x) \geq f(\beta) + L \cdot (\beta - x) \quad \forall x \in [\alpha, \beta]. \end{aligned}$$

En consecuencia, $(x, f(x)) \in \mathcal{R}$ para cualquier $x \in [\alpha, \beta]$, donde \mathcal{R} es un romboide cerrado delimitado por las funciones lineales anteriores, que son paralelas dos a dos. Esto es, el movimiento de la función queda restringido a dicho romboide. Los puntos de referencia del mismo serán los puntos de corte entre las funciones lineales, habiendo cuatro en total. Dos de ellos son $(\alpha, f(\alpha))$ y $(\beta, f(\beta))$, mientras que los otros dos se pueden determinar analíticamente. El recinto descrito aparece en un tono verde apagado en la Figura 4.2. Conociendo esta información, podemos determinar la longitud de la base y de la altura de \mathcal{R} , y calcular así su área. Nótese además que si $(\alpha, f(\alpha))$ y $(\beta, f(\beta))$ se encuentran en una misma recta de pendiente L , entonces solo habrá dos puntos de corte distintos entre las funciones lineales, por lo que f se comportará linealmente en $[\alpha, \beta]$ y \mathcal{R} tendrá medida nula.

Por otro lado, la gráfica de la función constantemente igual a $\frac{f(\alpha) + f(\beta)}{2}$ no estará contenida en su totalidad en el recinto descrito, excepto si $f(\alpha) = f(\beta)$. Por tanto, al calcular su distancia con f en $\mathcal{L}^1([\alpha, \beta])$, hemos de contabilizar un exceso fijo, y que viene dado por los dos triángulos simétricos T_1 y T_2 de color verde claro en la Figura 4.2. Al ser estos simétricos, su medida es igual que la de un rectángulo T . El área de este rectángulo también se puede calcular de forma analítica.

Ahora bien, en el peor de los casos, la función f corresponderá con alguna de las dos "funciones límite" que se señalan en color morado en la Figura 4.2, coincidiendo su gráfica con los lados superiores e inferiores del romboide \mathcal{R} , respectivamente. En este caso, el área de la diferencia entre f y la función constante viene dada por la mitad del romboide \mathcal{R} y el rectángulo T completo. Sabemos pues que

$$\left\| f - \frac{f(\alpha) + f(\beta)}{2} \right\|_{\mathcal{L}^1([\alpha, \beta])} \leq \frac{1}{2} \cdot \mu_2(\mathcal{R}) + \mu_2(T). \quad (4.1)$$

Conocida una sucesión de pares $\{(x_i, f(x_i))\}_{i=1}^k$, $k \in \mathbb{N}$, de forma que $x_1 = a$ y $x_k = b$ y

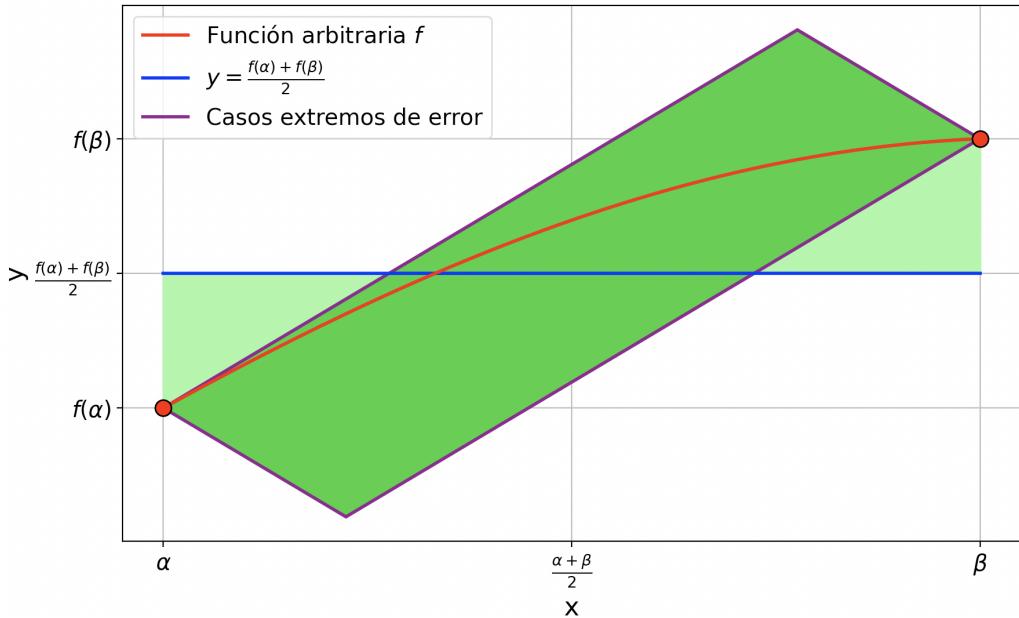


Figura 4.2: Gráfica que ilustra el máximo error en $\mathcal{L}^1([\alpha, \beta])$ entre una función f lipschitziana que pasa por los puntos rojos $(\alpha, f(\alpha))$ y $(\beta, f(\beta))$ y la función constantemente igual a $\frac{f(\alpha)+f(\beta)}{2}$ indicada en azul. El romboide de color verde apagado indica el recinto acotado de movimiento de f , determinado por su constante de Lipschitz $L > 0$. Los triángulos de color verde claro muestran las áreas fijas de diferencia de f con respecto de la función constante. Las líneas moradas representan las "funciones límite" que maximizan el error en $\mathcal{L}^1([\alpha, \beta])$ con respecto de la función constante.

$x_{i-1} < x_i \forall i \in \Gamma_k$, es posible acotar el error en entre f y $\frac{f(x_{i-1})+f(x_i)}{2}$ en $\mathcal{L}^1([x_{i-1}, x_i])$ para todo $i \in \Gamma_k$. En consecuencia, el error entre f y $S := \sum_{i=2}^k \frac{f(x_{i-1})+f(x_i)}{2} \cdot \chi_{[x_{i-1}, x_i]}$ en $\mathcal{L}^1(I)$ también estará acotado, y se puede calcular sin más que sumar los máximos errores del tipo (4.1) para cada uno de los subintervalos. Nótese que teniendo k datos, la función simple construida tendrá $k - 1$ términos.

En la Figura 4.3, se muestra un ejemplo del cálculo del error máximo entre una función lipschitziana de cuya gráfica se conocen un número concreto de puntos distribuidos aleatoriamente, y la función simple construida. Es fácil deducir que cuanto mayor sea la constante de Lipschitz considerada, mayores serán las áreas de error, puesto que la función tendrá más libertad de movimiento entre dos puntos cualesquiera.

4.2.2. Algoritmo adaptativo para la minimización del error

Una vez que hemos visto cómo se calcula el error en unas condiciones concretas, vamos a plantear un algoritmo que trate de minimizarlo de acuerdo a los datos que poseemos. Se tratará de un algoritmo adaptativo cuyo criterio de parada puede enfocarse de distintas maneras:

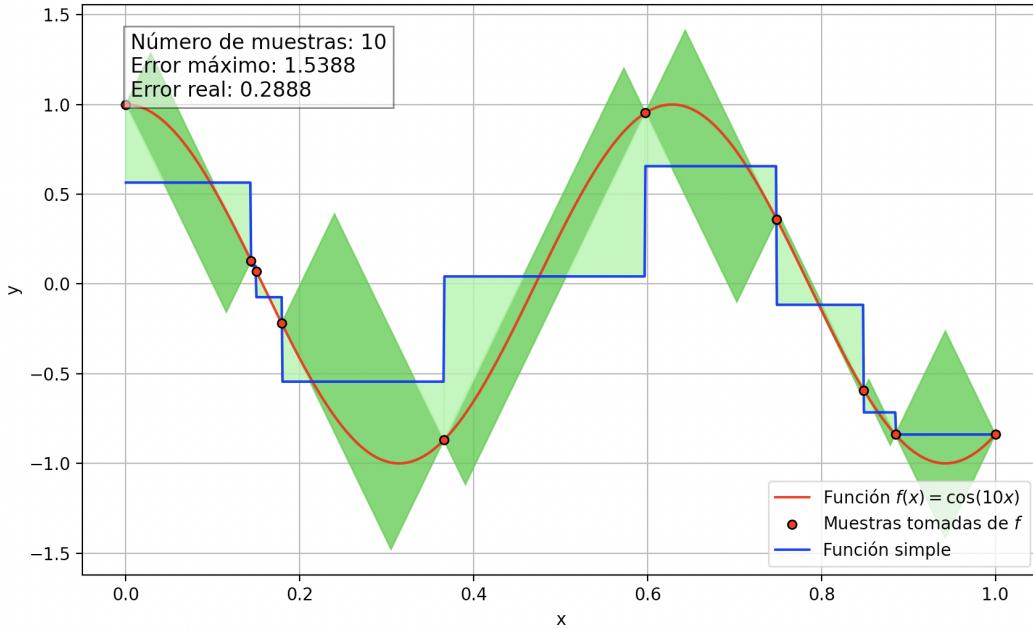


Figura 4.3: Gráfica que muestra la construcción de una función simple a partir de una muestra de $k = 10$ datos tomados de la función $f : [0, 1] \rightarrow \mathbb{R}$ con $f(x) = \cos(10x) \forall x \in [0, 1]$. Esta función es lipschitziana con constante $L = 10$, pues su derivada está acotada por $[-10, 10]$. Se calcula el error máximo bajo la suposición de que $L = 10$, sin conocimiento del comportamiento de la función fuera de los puntos seleccionados. Observamos que el error real (la distancia en $\mathcal{L}^1([0, 1])$ entre f y la función simple) es, en este caso, bastante más pequeño que el máximo calculado.

- **Acotar el error por debajo de cierto umbral:** El objetivo es encontrar una función simple que garantice una distancia con la función original menor que cierto $\epsilon > 0$ en $\mathcal{L}^1(I)$. Es posible que la cantidad de datos disponibles y/o su distribución en el espacio impidan verificar este criterio. Por tanto, en caso de que se hayan agotado los datos a analizar, también se finalizaría.
- **Encontrar un conjunto de datos limitado que ofrezca una buena aproximación:** En este caso, se busca mejorar la aproximación a la función original usando un número fijo de datos. Recordemos que el tamaño de la partición está ligado con la anchura/profundidad de la red neuronal que luego implementaremos, luego conviene que este número sea relativamente reducido.

Se puede optar por un criterio que mezcle los dos anteriores: por ejemplo, parar si se alcanza una cota determinada de error, o bien si ya se está usando un cierto número de datos.

El algoritmo toma su inspiración en algunos de los expuestos en [10], especialmente en el referente a la integración numérica de funciones lipschitzianas, pero es modificado para nuestro problema concreto. El pseudocódigo del mismo se expone en el [algoritmo 1](#).

Algoritmo 1: Minimización adaptativa del error dados unos datos

Input:

- **datosDisponibles:** Lista de datos conocidos, cada uno de la forma $[x, y]$, ordenados según su primera componente.
- **L:** Constante de Lipschitz.

Output:

- **datosElegidos:** Lista de datos elegidos de entre los disponibles para realizar la aproximación, también ordenados por la primera componente.
- **errorTotal:** Error máximo total calculado.

```

1 begin
2     // Inicialización
3     datosElegidos ← primer y último dato de datosDisponibles;
4     Eliminar primer y último dato de datosDisponibles ;
5     areasError ← {} ;           // Cola con prioridad para áreas de error.
6     errorTotal ← errorMaximo (datosElegidos[0], datosElegidos[1], L);
7     Insertar [0, errorTotal] en areasError;
8     // Proceso iterativo de minimización del error
9     while no se cumpla el criterio de parada do
10    [j, errorActual] ← Primer elemento de areasError;
11    mejorError1, mejorError2 ← 0, 0;
12    mejorError ← errorActual;
13    mejorDato ← Ninguno;
14    // Búsqueda de un dato que minimice el error en la ventana actual.
15    for nuevoDato in datosDisponibles do
16        if datosElegidos[j][0] < nuevoDato[0] < datosElegidos[j+1][0] then
17            nuevoError1 ← errorMaximo (datosElegidos[j], nuevoDato, L);
18            nuevoError2 ← errorMaximo (nuevoDato, datosElegidos[j+1], L);
19            nuevoError = nuevoError1 + nuevoError2;
20            if nuevoError < mejorError then
21                mejorError1, mejorError2 ← nuevoError1, nuevoError2;
22                mejorError ← nuevoError;
23                mejorDato ← nuevoDato;

24    // Actualización en caso de poder mejorar el error.
25    if hay mejorDato then
26        Insertar mejorDato en la posición j+1 de datosElegidos;
27        Eliminar mejorDato de datosDisponibles ;
28        Ajustar índices en la primera componente de areasError;
29        Insertar [j, mejorError1] en areasError ;
30        Insertar [j + 1, mejorError2] en areasError ;
31        errorTotal ← errorTotal - errorActual + mejorError;

32    return datosElegidos, errorTotal

```

Como observamos, se utiliza una **cola con prioridad** `areasError` que contiene elementos de la forma `[j, error]`, donde `j` indica el índice de la lista `datosElegidos` donde se encuentra un dato, y `error` es el máximo error calculado usando este y el siguiente dato de la lista, el cual se obtiene mediante la función `maximoError`, cuya forma se detalla en la [Subsección 4.2.1](#). El criterio para el orden en la cola es, precisamente, el del máximo error. Esta estructura se utiliza de la siguiente manera:

1. Inicialmente, se calcula el error utilizando $(a, f(a))$ y $(b, f(b))$, es decir, los puntos de la gráfica de la función f en los extremos el intervalo de trabajo $I = [a, b]$.
2. Se escoge el dato intermedio $(c, f(c))$, con $a < c < b$, que dé lugar al menor cómputo general de error entre a y b . Ahora, se introducen en la cola dos elementos, asociados a los errores calculados usando $(a, f(a))$ y $(c, f(c))$, y $(c, f(c))$ y $(b, f(b))$, respectivamente.
3. Se extrae el primer elemento de la cola, es decir, el área de mayor error entre dos datos, y se repite el segundo paso para esa ventana. El proceso se realiza sucesivamente, hasta que se cumpla el criterio de parada.
4. Si en algún momento el error es máximo en un subintervalo pero no existen más datos en su interior, entonces simplemente se extrae el error de la cola y se procede a intentar minimizar el siguiente.

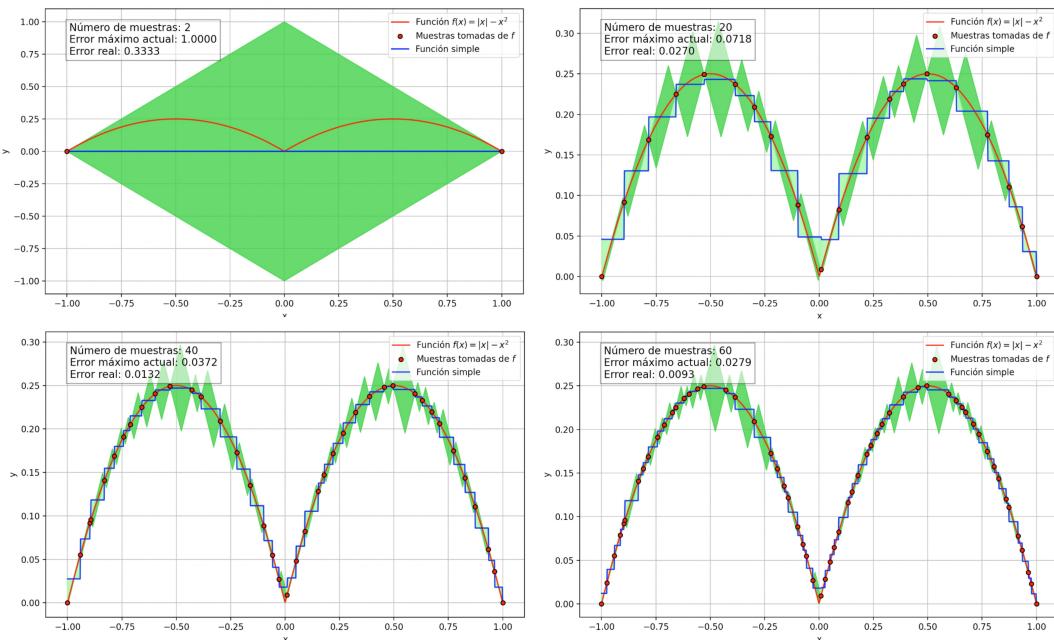


Figura 4.4: Evolución del algoritmo para un conjunto de $k = 100$ muestras disponibles de la forma $(x, f(x))$, con $f : [-1, 1] \rightarrow \mathbb{R}$ tal que $f(x) = |x| - x^2 \forall x \in [-1, 1]$, de forma que podemos considerar la constante $L = 1$. En este caso, se generan puntos según una distribución uniforme en $[-1, 1]$ y se construyen las muestras aplicándoles la función f . El algoritmo busca, en la medida de lo posible, reducir las máximas áreas de error presentes entre dos puntos.

El algoritmo tiende a reducir el error general de manera eficiente, tratando de considerar los datos más significativos, de manera que la pérdida se minimice usando el mínimo número posible de ellos. En la [Figura 4.4](#), se ofrece una visualización del funcionamiento del mismo. En este caso concreto, se observa cómo hay áreas relativamente grandes de error que no pueden reducirse, lo cual ocurre por no existir datos intermedios disponibles.

En caso de existir un cierto ruido en los datos, lo cual es usual en problemas reales, se podría adaptar el algoritmo con técnicas que se explican en [\[10\]](#). En nuestro caso, nos limitaremos a estudiar esta versión.

4.3. Implementación de las redes

Llegados a este punto, disponemos de herramientas para construir funciones simples que aproximen funciones lipschitzianas hasta cierto nivel de exactitud a partir de unos datos. Para constantes de Lipschitz reducidas, grados de precisión holgados y datos que no tengan demasiada distancia entre ellos, el número de términos de las mismas será manejable y permitirá la construcción de arquitecturas neuronales eficientes siguiendo las claras instrucciones que se proveen en la [Sección 3.1](#) y la [Sección 3.2](#). Las únicas consideraciones adicionales que hemos de tener en cuenta son las siguientes:

- **Peso relativo de las aproximaciones:** El argumento de la construcción es que se puede aproximar en primer lugar la función original mediante una función simple, y posteriormente la función simple mediante una red neuronal. Para asegurar que el error total de la red neuronal con respecto de la función original es menor que cierto $\epsilon > 0$, debemos garantizar que la suma de ambos errores queda por debajo de ϵ . Esto se puede lograr asignando un peso relativo a cada aproximación. Si se da mayor trascendencia a la segunda, la función resultante de la red se parecerá más a una función escalonada. Si se da mayor peso a la primera, las transiciones serán menos bruscas, pero a costa de un aumento en el número de términos de la función simple, y por tanto del tamaño de la arquitectura. Ambos enfoques son válidos si buscamos una red que aproxime en $\mathcal{L}^1(I)$. Por ejemplo, en la demostración del [Teorema 3.3](#), se opta por asignar el mismo peso a ambos factores. En principio, se escogerá este mismo enfoque.
- **Cálculo del factor $\delta > 0$:** Según vimos, este factor determina qué tanto se aproxima cada término de la función simple mediante la red, usando el criterio de la distancia en $\mathcal{L}^1(I)$. Un posible método heurístico para calcularlo es un algoritmo iterativo que vaya reduciendo el valor de δ hasta que, efectivamente, obtengamos el grado de aproximación deseado en cada subintervalo, de forma que el cómputo de error total con respecto de la función simple completa sea suficientemente pequeño. Podemos rebajar este valor tanto como deseemos.

Poniendo en la práctica estas observaciones, podemos diseñar redes neuronales que realizan funciones como las de la [Figura 4.5](#). En ambos casos, se aproxima la misma función simple, calculada según el [algoritmo 1](#), usando un total de $k = 44$ datos muestreados de una función concreta, que resultan en una partición de $m = 43$ subintervalos en total. Por ello, las características de estas dos arquitecturas específicas son:

- En la zona superior: Red con funciones de activación sigmoides, una única capa oculta ($d = 2$) y anchura $w = 2 \cdot 43 = 86$, lo cual implica únicamente 258 parámetros.

- En la zona inferior: Red ReLU con anchura $w = 1 + 4 = 5$ y profundidad $d = 4 \cdot 43 + 2 = 174$, con un número total de parámetros igual a 5,108.

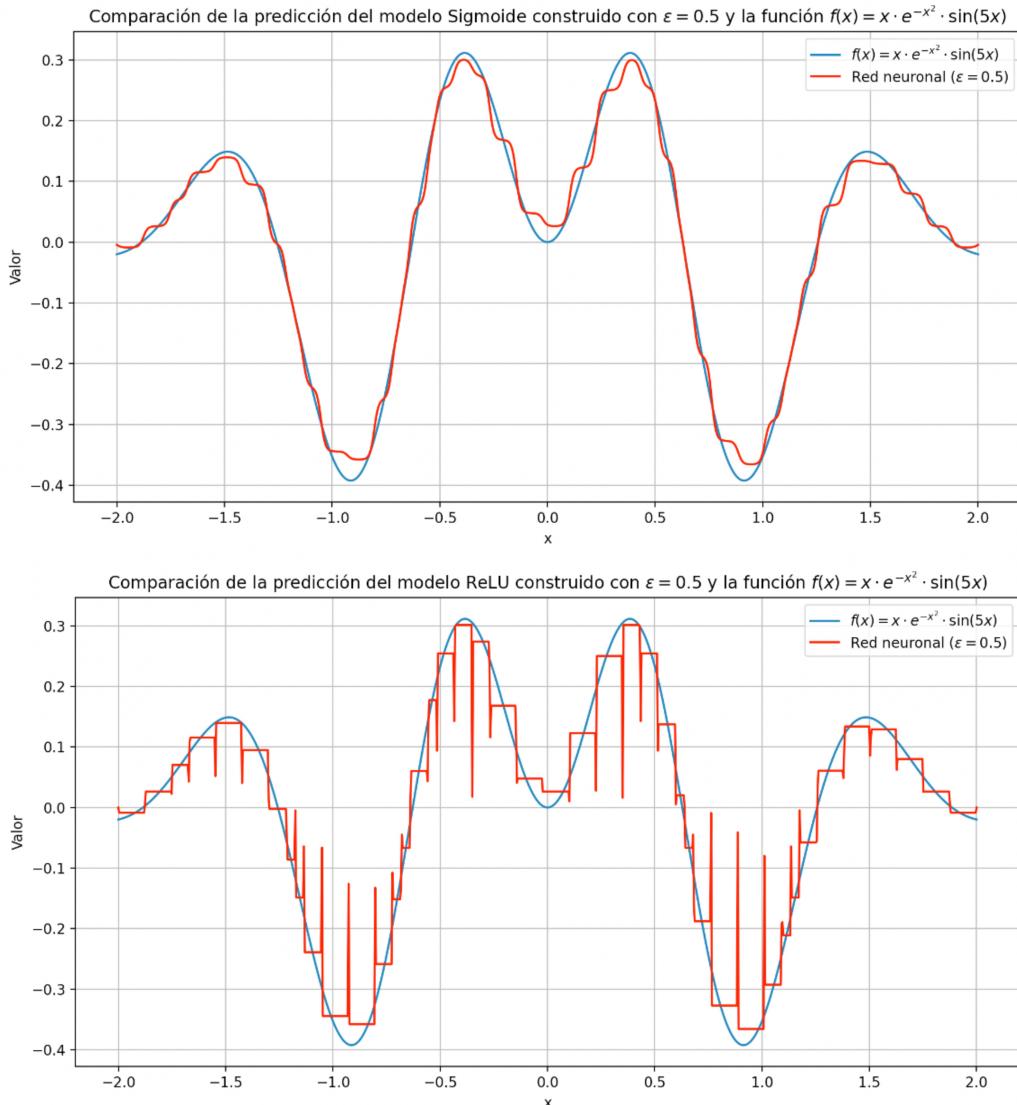
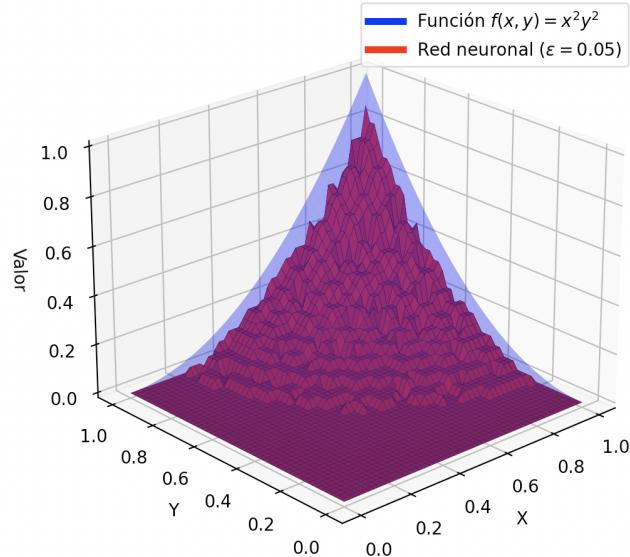


Figura 4.5: En la parte inferior, se muestra la función realizada por una red neuronal ReLU de anchura acotada cuya distancia en $L^1([-2,2])$ con la función f dada por $f(x) = x \cdot e^{-x^2} \cdot \sin(5x) \forall x \in [-2,2]$ es menor que $\epsilon = 0.5$. En la parte superior, se muestra la función realizada por una red con funciones de activación sigmoide y una sola capa oculta que verifica la misma condición. Se aprecia como los picos de esta última son más suaves.

Observación 4.3.1. Aunque sólo se muestren ejemplos de funciones de variable única, el código implementado para redes ReLU acotadas en anchura está preparado para aproximar funciones simples definidas con respecto de intervalos n -dimensionales, $n > 1$, provisto un

conjunto de estos que se haya obtenido, por ejemplo, mediante una extensión del algoritmo explicado en la [Sección 4.2](#) o similares. En la [Figura 4.6](#) se observa una visualización que ilustra este hecho.

Comparación de la predicción del modelo ReLU construido con $\epsilon = 0.05$ y la función $f(x, y) = x^2y^2$



[Figura 4.6](#): Función realizada por una red neuronal ReLU totalmente conectada y acotada en anchura, que aproxima la función dada por $f(x, y) = x^2y^2 \forall (x, y) \in [0, 1]^2$, con $\epsilon = 0.05$, en $\mathcal{L}^1([0, 1]^2)$. En este caso, se han obtenido los términos de la función simple mediante fuerza bruta, por lo que el tamaño de la red generada es extremadamente grande.

Al estudiar la arquitectura de la [Sección 3.2](#), no es difícil notar que, de alguna manera, esta se puede disponer "en vertical", generando un modelo de anchura $w = n + 2 \cdot m$ y profundidad $d = 4 \cdot n + 1$, donde n es el tamaño de la entrada y m es el número de términos de la función simple. Para ello, basta con colocar la neurona $(n + 3)$ -ésima y $(n + 4)$ -ésima de cada capa de los bloques neuronales descritos en la [Figura 3.3](#) en paralelo, de forma que se vayan computando las aproximaciones de las funciones características de manera concurrente en cada capa. Las coordenadas del input se pueden traspasar mediante n neuronas únicas a través de las capas, nutriendo a las neuronas que calculan las funciones características con sus valores cuando sea necesario. Se prescinde del sistema que almacena sumas parciales descrito en la [Figura 3.5](#), ya que en este caso podemos combinar linealmente todas las salidas de los bloques que calculan cada función característica mediante una última capa lineal. Para mayor detalle de esta reestructuración, se puede examinar el código provisto.

La función que realiza esta arquitectura dispuesta en vertical es exactamente la misma que la que realiza la arquitectura ReLU acotada en anchura, para una parametrización establecida. Se decide considerar esta configuración ya que resulta más natural: la profundidad de la red será como mínimo 5, con toda la potencialidad que eso conlleva y que se refleja en numerosos recursos, como [\[59, 18\]](#). Además, en nuestra restricción a funciones lipschitzianas, el factor m será relativamente reducido, por lo que el tamaño global de la red será razonable.

En resumen, hemos implementado tres arquitecturas distintas, las cuales para el caso unidimensional, tendrán las siguientes características:

- **Red Sigmoidal:** Anchura $w = 2 \cdot m$ y profundidad $d = 2$.
- **Red ReLU horizontal:** Anchura $w = 5$ y profundidad $d = 4 \cdot m + 2$.
- **Red ReLU vertical:** Anchura $w = 2 \cdot m + 1$ y profundidad $d = 5$.

Las tres son capaces de aproximar cualquier función lipschitziana provisto un valor $m \in \mathbb{N}$ apropiado.

4.4. Estudio de la parametrización inicial

En esencia, lo que hemos hecho en el apartado anterior es implementar una red que, con unos pesos y biases determinados, aproxima una función a partir de unos datos asociados a ella. Además, la hemos producido de manera totalmente analítica: hemos establecido los parámetros de manera explícita a sabiendas de que verificarán directamente la condición de proximidad con respecto de la función que representa a los datos, sin necesidad de aplicar un proceso iterativo de entrenamiento de ningún tipo.

Ahora, cabe preguntarse si el clásico método de programación hacia atrás junto con Descenso de Gradiente, que se expone en la Subsección 2.3.2, puede mejorar aún más los resultados de aproximación, partiendo de estos parámetros iniciales que ya ofrecen errores relativamente reducidos. Paralelamente, el desarrollo de los siguientes experimentos servirá de base para obtener algunas reflexiones sobre la importancia de la anchura y la profundidad de las redes neuronales, y su capacidad de generalización a nuevos datos.

4.4.1. Procedimiento experimental

Todos las ejecuciones del experimento se llevarán a cabo bajo las siguientes premisas:

- Se establece una **función objetivo lipschitziana** $f : I \rightarrow \mathbb{R}$, con I un intervalo de \mathbb{R} y constante de Lipschitz $L = 1$.
- Se fijan tres subconjuntos de valores de entrada contenidos en el intervalo I . Los tres se obtienen mediante distribuciones uniformes en I , fijando semillas distintas en cada caso. A cada valor, se le aplica la función f . Así, construimos tres subconjuntos de datos etiquetados: el de **entrenamiento**, el de **validación** y el de **evaluación**. Los tamaños de los mismos se fijarán en 100, 25 y 25, respectivamente.
- Se ejecuta el **algoritmo 1** sobre el conjunto de entrenamiento, utilizando como criterio de parada el uso de un número determinado de datos. Se utilizarán 10, 30, 50 y 100 datos en la ejecución de este algoritmo para cada función. Nótese que para lograr una aproximación de f en $\mathcal{L}^1(I)$, en el conjunto de entrenamiento han de estar los pares $(a, f(a))$ y $(b, f(b))$ donde $a, b \in \mathbb{R}$, $a < b$, son los extremos del intervalo I .

Así, obtenemos una función simple con un número de términos concreto que minimiza el error con respecto de f en $\mathcal{L}^1(I)$, según los datos de los que disponemos. Operamos sobre el conjunto de entrenamiento ya que los cálculos afectarán directamente a los pesos y biases de la posterior red neuronal, y querremos asegurarnos de que la red generaliza correctamente mediante los conjuntos de validación y test.

- Se construye una arquitectura parametrizada explícitamente usando la función simple anterior. Se es más laxo en este caso a la hora de determinar el parámetro $\delta > 0$ de estas arquitecturas, pues no se busca alcanzar un grado de aproximación concreto inicialmente, sino simplemente una buena aproximación de la que partir. En el caso de la(s) red(es) ReLU, se toma una proporción del subintervalo más pequeño de la partición (por ejemplo, podemos tomar el 30 % del mismo). En el caso de la red Sigmoide, simplemente se sitúa en $\delta = 0.01$, tras varias pruebas. En la práctica, conviene que este valor no sea demasiado pequeño para evitar problemas relacionados con la estabilidad numérica durante el proceso de entrenamiento.
- Se entrena dicha arquitectura a partir de los parámetros iniciales calculados, y por otro lado, se entrena usando parámetros iniciales aleatorios. En el segundo caso, los pesos se inicializan utilizando la estrategia de Xavier Glorot [20], y los biases se toman como ceros.
- Para uniformizar, en todos los casos se utilizará:
 - **Optimizador Adam:** Se utilizan los hiperparámetros por defecto: tasa de aprendizaje inicial igual a 0.001, factor $\beta_1 = 0.9$ y factor $\beta_2 = 0.999$. Para más información de su significado, se ha de consultar [27].
 - **Función de pérdida:** Error Cuadrático Medio (MSE).
 - **Tamaño de lote:** Igual a 16. Se selecciona con respecto de otros estándares como 32 o 64 por el reducido tamaño del conjunto de entrenamiento.
 - **Número de épocas:** Un total de 15. Esto será suficiente en la mayoría de los casos, previniendo así el sobreajuste.
- La comparación del rendimiento de los modelos se hará utilizando dos **métricas** sobre el conjunto de entrenamiento y el de validación: el propio **MSE** y el **coeficiente de determinación (R^2)**, que se explica en la [Subsubsección 2.3.3.1](#). Así mismo, se examinarán las gráficas de convergencia, en la búsqueda de conclusiones sobre generalización y estabilidad. Las instancias del conjunto de test solo se predecirán con el mejor de los modelos, evitando así incurrir en la práctica del *data snooping* [62].

4.4.2. Arquitecturas empleadas

En primer lugar, apreciamos en la [Tabla 4.1](#) el tamaño de las redes neuronales consideradas, dependiendo del número de datos k que tengamos en cuenta a la hora de implementarlas.

	$k = 10$			$k = 30$			$k = 50$			$k = 100$		
	w	d	p	w	d	p	w	d	p	w	d	p
Sigmoide	18	2	54	58	2	174	98	2	294	198	2	594
ReLU horizontal	5	38	1,027	5	118	3,427	5	198	5,827	5	398	11,827
ReLU vertical	19	5	1,197	59	5	10,797	99	5	29,997	199	5	119,997

Tabla 4.1: Tamaños de las arquitecturas según el número de datos k que se considere para construirlas con el planteamiento propuesto. w indica la anchura, d la profundidad y p el número de parámetros entrenables (pesos y biases) de cada tipo de red.

Observamos que las configuraciones obtenidas son de muy distinta índole. Además, una red de anchura w y profundidad d , que toma una entrada y produce una única salida, podrá tener hasta $p = (d - 2) \cdot w^2 + (d + 1) \cdot w$ parámetros. Por lo tanto, al fijar d , el aumento de w conduce a un mayor incremento en el número de parámetros p en comparación del aumento de d para w fijo. Notemos que en el caso de ReLU horizontal, hay menos parámetros que los que estipula la fórmula, ya que en nuestra construcción, sus primeras 4 capas tienen tan solo 3 neuronas cada una en lugar de 5.

4.4.3. Resultados obtenidos

Pasamos ahora a aplicar el procedimiento de entrenamiento explicado tomando como objetivo la sencilla función $f : [-3, 3] \rightarrow \mathbb{R}$ dada por $f(x) = \sin(x) \quad \forall x \in [-3, 3]$, que es lipschitziana con constante $L = 1$. Los resultados detallados figuran en la [Tabla 4.2](#). Se muestra el valor de las métricas medidas sobre los conjuntos de entrenamiento y validación tras la última época, para cada una de ellas, se miden las métricas utilizando los pesos iniciales explícitos (sin entrenamiento previo), empleando los parámetros obtenidos tras entrenar a partir de los explícitos, y los calculados tras el entrenamiento empezando con pesos/biases iniciales aleatorios.

4.4.4. Interpretación de las métricas

Pasamos ahora a comentar una serie de aspectos numerados que se derivan del análisis de la [Tabla 4.2](#):

1. Fijémonos en primer lugar en los **tiempos** requeridos para el entrenamiento. Si bien son todos bastante reducidos debido al escaso número de épocas y la poca cantidad de datos, notamos que en todos los casos los modelos del tipo ReLU horizontal tardan más en entrenarse. Esto contrasta con el hecho de que para cada valor k , ReLU vertical tiene más parámetros, por lo que tendremos que optimizar una función de pérdida con más entradas en este caso.

La explicación de este hecho se encuentra en el funcionamiento de la propagación hacia atrás, ya que se requieren más recursos para almacenar los cálculos intermedios en cada capa y se realizan más operaciones tanto en la fase hacia adelante como en la fase hacia atrás, lo que conlleva una dilatación del tiempo en las sucesivas actualizaciones de pesos.

Por otro lado, los tiempos en las redes Sigmoide y ReLU vertical se mantienen bastante estables a pesar del aumento en la anchura de las arquitecturas. Por tanto, concluimos que la profundidad es el criterio que tiene el mayor peso a la hora de determinar el tiempo estimado de entrenamiento de un modelo.

2. Notemos que a medida que aumentamos k , las métricas calculadas sobre el conjunto de entrenamiento en los modelos ReLU base tienden a ser peores, pero las del conjunto de validación tienden a mejorar. Resulta extraño observar cómo, por ejemplo, cuando $k = 100$, se mide un coeficiente R^2 igual a cero para los datos de entrenamiento y de 0.9988 para el conjunto de validación.

La explicación de esto se halla en la forma de la función realizada por el modelo tras su construcción, que se puede observar con claridad en la parte inferior de la [Figura 4.5](#).

4 Construcción Explícita de Redes Neuronales

		k = 10				k = 30					
		MSE		R ²		T (s)	MSE		R ²		
		Entr.	Val.	Entr.	Val.		Entr.	Val.	Entr.	Val.	
Sigmoide	Base	0.0122	0.0161	0.9767	0.9701	-	0.0005	0.0008	0.9989	0.9984	-
	Explícitos	0.0096	0.0131	0.9816	0.9756	1.72	0.0003	0.0008	0.9992	0.9983	1.69
	Aleatorios	0.2464	0.2445	0.5327	0.5479	1.52	0.2070	0.2004	0.6074	0.6295	1.38
ReLU horizontal	Base	0.1277	0.1087	0.7578	0.7990	-	0.1598	0.0289	0.6969	0.9464	-
	Explícitos	0.0074	0.0070	0.9858	0.9869	8.73	0.0342	0.0528	0.9351	0.9022	16.20
	Aleatorios	0.5276	0.5444	-0.0005	-0.0061	7.73	0.5279	0.5452	-0.0011	-0.0077	16.87
ReLU vertical	Base	0.1277	0.1087	0.7578	0.7990	-	0.1598	0.0289	0.6969	0.9464	-
	Explícitos	0.0045	0.0149	0.9912	0.9724	2.65	0.0086	0.0608	0.9835	0.8875	1.99
	Aleatorios	0.1127	0.0977	0.7862	0.8193	2.29	0.0853	0.0745	0.8382	0.8622	1.85

		k = 50				k = 100					
		MSE		R ²		T (s)	MSE		R ²		
		Entr.	Val.	Entr.	Val.		Entr.	Val.	Entr.	Val.	
Sigmoide	Base	0.0001	0.0003	0.9996	0.9992	-	0.0001	0.0003	0.9997	0.9993	-
	Explícitos	0.0000	0.0003	0.9998	0.9993	1.57	0.0000	0.0003	0.9998	0.9993	1.29
	Aleatorios	0.1749	0.1615	0.6682	0.7013	1.53	0.1722	0.1542	0.6733	0.7149	1.24
ReLU horizontal	Base	0.2519	0.0339	0.5222	0.9372	-	0.5273	0.0006	0.0000	0.9988	-
	Explícitos	0.0628	0.0667	0.8808	0.8765	30.38	0.0680	0.0725	0.8710	0.8659	65.83
	Aleatorios	0.5280	0.5453	-0.0012	-0.0079	34.04	0.5276	0.5443	-0.0005	-0.0059	59.32
ReLU vertical	Base	0.2519	0.0339	0.5222	0.9372	-	0.5273	0.0006	0.0000	0.9988	-
	Explícitos	0.0198	0.0295	0.9624	0.9454	2.11	0.0238	0.0616	0.9548	0.8860	1.88
	Aleatorios	0.0389	0.0345	0.9262	0.9361	2.09	0.0323	0.0316	0.9386	0.9414	1.71

Tabla 4.2: Resultados del experimento para arquitecturas neuronales de diversos tamaños, considerando el modelo entrenado partiendo tanto de parámetros determinados previamente (Explícitos) como de parámetros aleatorios (Aleatorios), así como el modelo de parámetros inicial sin entrenar (Base). Se mide el error cuadrático medio (MSE) y el coeficiente de determinación (R^2) con respecto de los conjuntos de entrenamiento (Entr.) y validación (Val.) tras la última época, además del tiempo consumido por el entrenamiento en segundos (T). El objetivo es la función senoidal definida en $[-3, 3]$. Los valores de las métricas están truncados a cuatro decimales, y los del tiempo a dos.

Recordemos que la aproximación se construye a partir de datos de entrenamiento, y en tal caso, estos van a caer exactamente en las ventanas donde la aproximación ReLU de cada término simple falla. Esto es, en los picos que aparecen en la gráfica. En el caso $k = 100$, estamos usando todo el conjunto de entrenamiento para definir los parámetros del modelo, luego todos los datos se predecirán muy incorrectamente en primera instancia. Sin embargo, el conjunto de validación se genera aleatoriamente, y generalmente sus datos no caerán en esas ventanas, sino en áreas en las que la predicción es más correcta. Por tanto, al principio, la red neuronal falla para el conjunto de entrenamiento pero es bastante precisa en las predicciones para la mayoría de inputs que no están en él.

Con el posterior entrenamiento, conseguimos afinar la predicción en el conjunto de entrenamiento, a la par que mantenemos buenos resultados en validación. De manera un poco imprecisa, podemos intuir que el proceso de entrenamiento "moldea" la función de la red de forma que también prediga bien los datos de entrenamiento, manteniendo

buenas métricas en validación.

3. En las redes de tipo Sigmoide el fenómeno anterior no ocurre ya que la forma de aproximar es distinta, y los pronunciados picos correspondientes a cada término de la función simple se "cancelan" unos con otros, de forma que las predicciones en los extremos de los subintervalos (que corresponden a los datos de entrenamiento) no son tan desacertadas. Esto se pone de manifiesto en la parte superior de la [Figura 4.5](#).

No sorprende por tanto que, generalmente, sea este modelo el que ofrezca mejores resultados tras el entrenamiento con pesos iniciales explícitos. Al fin y al cabo, es el que parte de una mejor base, y el entrenamiento siempre mejora las métricas, al menos en el conjunto de entrenamiento, pues minimiza la función de pérdida para este.

4. Si bien el modelo de tipo Sigmoide ofrece buenos resultados tras el entrenamiento con parámetros iniciales determinados, no hace tan buen trabajo con los aleatorios. Observamos coeficientes de determinación que no pasan de 0.75 en ninguno de los casos. Si bien es cierto este que va aumentando ligeramente conforme se incrementa la anchura del modelo, parece que no lo hace al mismo ritmo ni en la misma magnitud que ReLU vertical. Lo que ocurre es que la restricción de la arquitectura a una única capa oculta aminora sustancialmente el potencial de la red para aprender representaciones internas de los datos, por lo que se produce el fenómeno de **infrajuste**: la adaptación a la función original que obtiene el algoritmo de optimización resulta insuficiente.
5. Por último, hemos de destacar un importante aspecto del modelo ReLU horizontal. En todos los casos, se observa una tendencia más o menos ascendente en las métricas conforme aumentamos el valor k que define la arquitectura concreta. Esto es lógico pues el mayor número de parámetros confiere una mayor expresividad a los modelos. Sin embargo, parece producirse un estancamiento cuando utilizamos parámetros iniciales aleatorios para entrenar ReLU horizontal. Para todos los valores de k , dicho modelo parece incapaz de aprender. No solo no generaliza, si no que no ajusta los datos de entrenamiento en absoluto (los coeficientes de determinación son negativos, lo cual es indicativo de que el modelo predice incluso peor que el que simplemente toma la media de los valores de las etiquetas).

Este hecho se puede achacar, evidentemente, a las poco ortodoxas características del modelo concreto. Generalmente, no se suelen plantear arquitecturas tan profundas y con tan pocas neuronas, ya que los pocos pesos presentes entre capas (tan solo 30 en nuestro caso) no son suficientes para aprender representaciones complejas de los datos ni establecer niveles de abstracción, como se esperaría de una red profunda. Esta cuestión, junto con la problemática de las neuronas ReLU muertas o **Dying ReLU**, que tiene un efecto muy notable debido a las pocas neuronas en cada capa, imposibilita el entrenamiento del modelo. Es más, existen estudios como [\[36\]](#) que demuestran que las redes de este tipo (profundas y estrechas) convergerán a estados erróneos de la media o la mediana de la función objetivo con alta probabilidad, incluso cuando se utilizan técnicas específicas, como la inicialización asimétrica de pesos. Esta última cuestión se aborda en artículos como [\[65\]](#).

Eso sí, con el enfoque de pesos iniciales explícitos, los resultados no son tan negativos, ya que arrancamos desde un estado muy favorable.

4.4.5. Interpretación de las curvas de aprendizaje

Ahora bien, para obtener conclusiones sólidas no basta con observar las métricas finales, sino que también hemos de analizar las curvas de aprendizaje. Ciertamente, si estudiamos las curvas para cada modelo y caso de inicialización de parámetros, estas se comportan de manera similar independientemente del tamaño considerado. Es por ello que analizamos las curvas para el caso en que $k = 100$, habiendo comprobado numéricamente que en general, es el que genera arquitecturas más potentes. Las gráficas correspondientes se encuentran en la Figura 4.7.

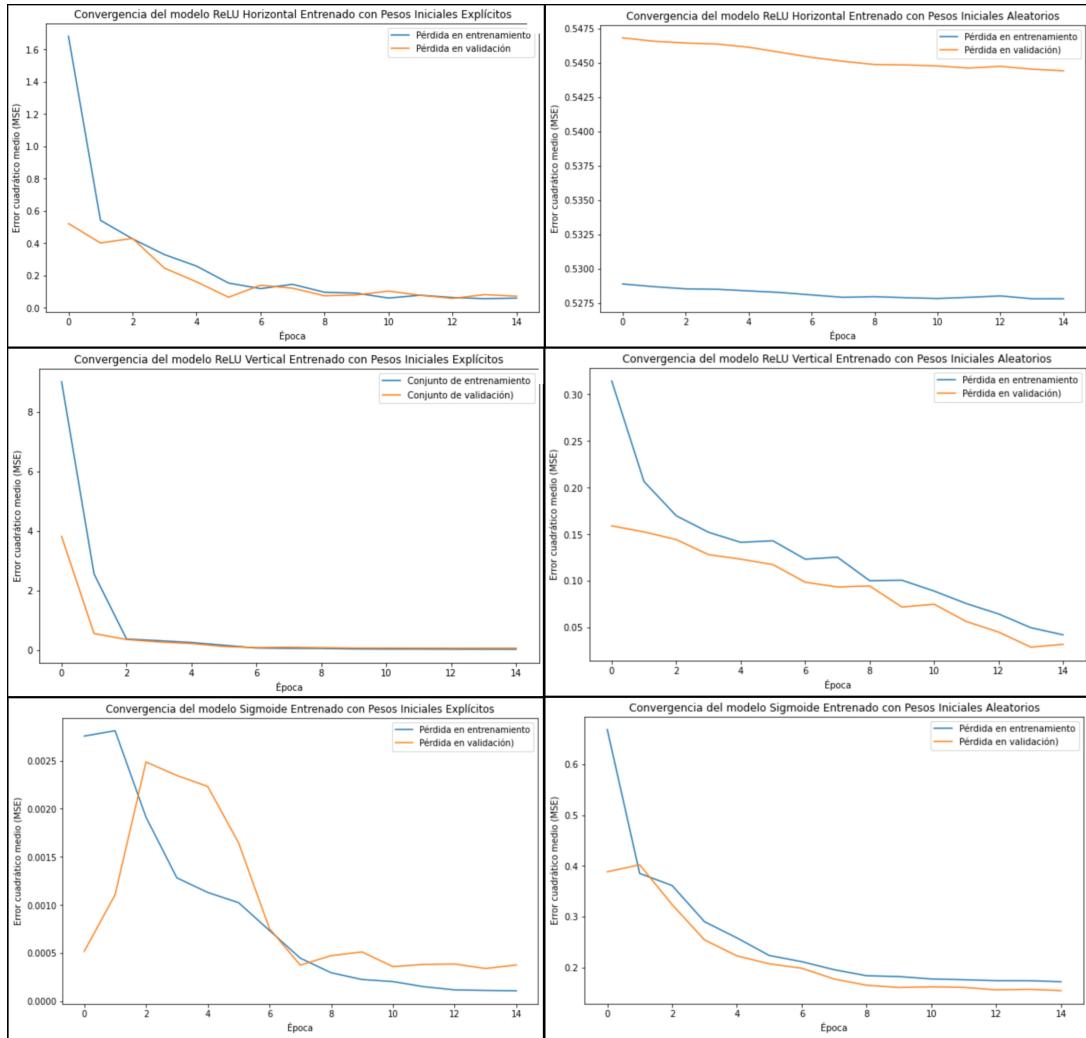


Figura 4.7: Curvas de aprendizaje de los modelos considerados para $k = 100$. Por filas, de arriba abajo, tenemos las curvas para el modelo ReLU Horizontal, el modelo ReLU Vertical y el modelo Sigmoidal. Por columnas, de izquierda a derecha, las curvas para pesos iniciales explícitos y pesos iniciales aleatorios.

Estas curvas nos indican la siguiente información:

- Los modelos ReLU entrenados con pesos iniciales explícitos parecen seguir el mismo

comportamiento: se produce una convergencia rápida de la pérdida en el entrenamiento, y a partir de entonces apenas hay mejora. La interpretación es que mediante este enfoque, estamos sesgando a la red de manera que caemos en mínimos locales mucho más rápido.

En cuanto a la validación, su convergencia es menos acusada, debido al hecho de que partimos de buenas predicciones en el modelo base, por la razón que apuntábamos en la [Subsección 4.4.4](#). Si bien es cierto que no mejora mucho sus resultados, al menos los mantiene.

- Para analizar la curva del modelo Sigmoide entrenado con pesos iniciales explícitos, hemos de fijarnos en la escala del MSE. Como vemos, se parte de un error muy bajo por las características del modelo base, luego la fluctuación en validación no es preocupante ni denota una inestabilidad en la convergencia. Lo cierto, aún así, es que la mejoría es muy reducida en términos cuantitativos.
- En la alternativa con pesos iniciales aleatorios para el modelo ReLU horizontal no se produce aprendizaje alguno, como se comentó, quedando este hecho patente en el aspecto de la gráfica.
- Por su parte, el modelo Sigmoide con pesos iniciales aleatorios sí que parece aprender de manera estable, pero su curva se va frenando a partir de la época 8 aproximadamente, quedando en valores de pérdida aún bastante altos, lo cual cuadra con la limitación derivada de la insuficiencia de parámetros entrenables que ya apuntábamos en el examen de las métricas de la última época.
- Por último, el entrenamiento del modelo ReLU vertical con pesos iniciales aleatorios, reduce significativamente el error cuadrático medio a través de las épocas, tanto de entrenamiento como de validación, de forma bastante uniforme. La tendencia, además, parece indicar que el modelo seguirá aprendiendo si se entrena un mayor número de épocas. Esto es factible dados los bajos tiempos de entrenamiento que presenta para 15 épocas.

4.4.5.1. Evaluación del modelo final

Aunque en el análisis anterior solo se muestran los resultados de la función seno definida en $[-3, 3]$, se ha repetido el experimento con otras funciones, obteniendo conclusiones análogas.

Es por ello que para culminar, vamos a evaluar el que hemos seleccionado como modelo más adecuado y robusto, es decir, el modelo **ReLU vertical entrenado con pesos iniciales aleatorios** para diversas funciones objetivo. Usaremos todos los datos del conjunto de entrenamiento generado para determinar la arquitectura de la red ($k = 100$) de forma que tendrá anchura $w = 199$, profundidad $d = 5$ y un total de $p = 119,997$ parámetros. En la [Figura 4.8](#), se muestran las métricas y visualizaciones de resultados para tres funciones objetivo, con respecto de los conjuntos de test que se obtienen a partir de ellas. El procedimiento de entrenamiento es el mismo en cada caso, pero esta vez realizamos un total de 20 épocas.

4.4.6. Conclusiones generales

Así, todo parece apuntar que si queremos un modelo robusto y que realmente aprenda características de los datos para poder realizar predicciones fiables, lo mejor es optar por una

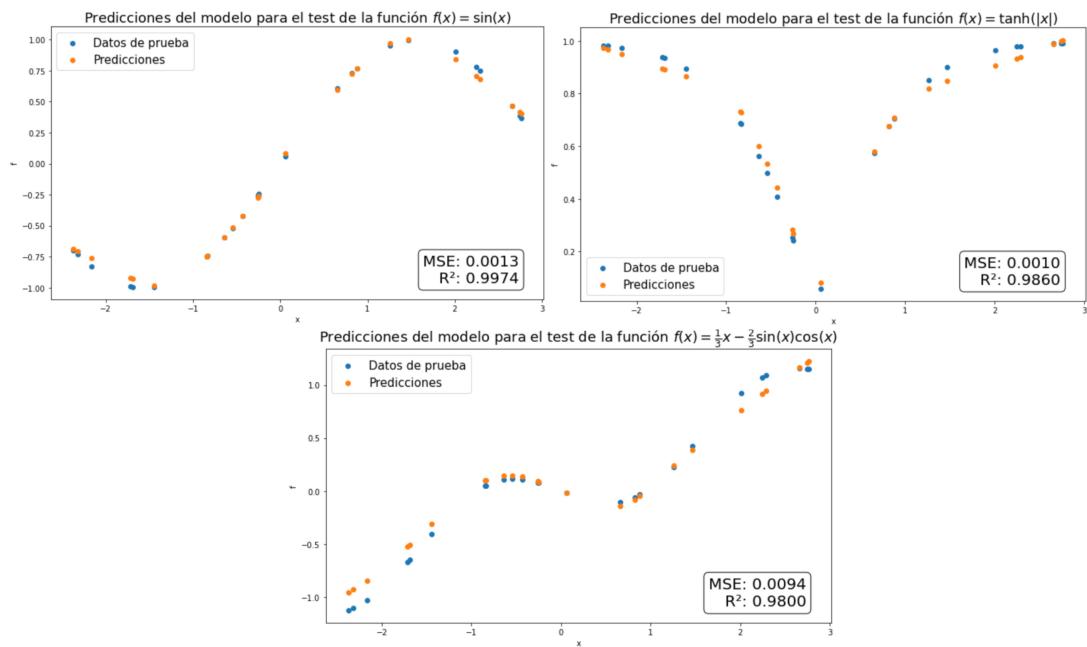


Figura 4.8: Evaluación del rendimiento del modelo ReLU vertical con respecto de un conjunto de test muestreado a partir de las funciones $x \mapsto \sin(x)$, $x \mapsto \tanh(|x|)$ y $x \mapsto \frac{1}{3}x - \frac{2}{3}\sin(x) \cdot \cos(x)$, todas ellas lipschitzianas con constante $L = 1$ y definidas en $[-3, 3]$.

arquitectura del tipo ReLU vertical. Es decir, en la práctica, lo más recomendable es buscar un **equilibrio entre profundidad y anchura**, evitando los enfoques extremos que acotan demasiado alguno de dichos factores, ya que pese a que poseen la capacidad expresiva para realizar un extenso repertorio de funciones, los métodos de optimización como el Descenso de Gradiente tienen dificultades para encontrar parámetros óptimos que les otorguen esa competencia utilizando pesos iniciales aleatorios.

En cuanto al enfoque de los pesos iniciales explícitos, si bien puede resultar interesante en condiciones controladas, no nos proporciona garantías de mejoría sustancial cuando lo encadenamos con un proceso de entrenamiento estándar. Estos pesos introducen un **sesgo** muy arbitrario en el procedimiento, forzando a caer de forma rápida en mínimos locales de la función de pérdida muy específicos, que podrían no ser los que proporcionan una mayor generalización y/o ajuste de los datos. Como ventajas de esta perspectiva, se puede argumentar que el procedimiento permite encontrar esos mínimos locales en un menor número de épocas (aunque podrían ser subóptimos), además de proporcionarnos cierta explicabilidad de cómo funciona la red internamente a la hora de realizar predicciones.

No obstante, lo más apropiado es dejar que el algoritmo, de forma autónoma, encuentre una manera de ajustar los pesos mediante la inicialización con parámetros aleatorios, que proveen una mayor libertad de movimiento dentro del espacio de soluciones. Eso sí, debemos entender que, en contraposición, este procedimiento es totalmente opaco, y no podremos explicar de manera sencilla en qué se basa exactamente el modelo para tomar decisiones. De ahí que en numerosas ocasiones se refiera a las redes neuronales como modelos de *caja negra*. La evidencia empírica nos dice que a pesar de ello, funcionan correctamente. Para profundizar en esta cuestión, se recomiendan artículos como [34].

5 Comentarios Finales

Para cerrar el trabajo, se incluyen una serie de aspectos relevantes sobre su desarrollo que se han de tener en cuenta.

5.1. Metodología, organización y planificación

Inicialmente, la propuesta del proyecto tenía una dirección clara. La piedra angular del mismo sería el artículo [37]. El planteamiento consistía en analizar con detalle algunos de los resultados matemáticos que en este se demuestran, y tratar de encontrar una vía satisfactoria para implementar y poner a prueba algunas de estas ideas utilizando las técnicas comunes del ámbito práctico. De esta forma, el desarrollo del proyecto se dividió en **cuatro grandes etapas**, que se representan en la [Figura 5.1](#):

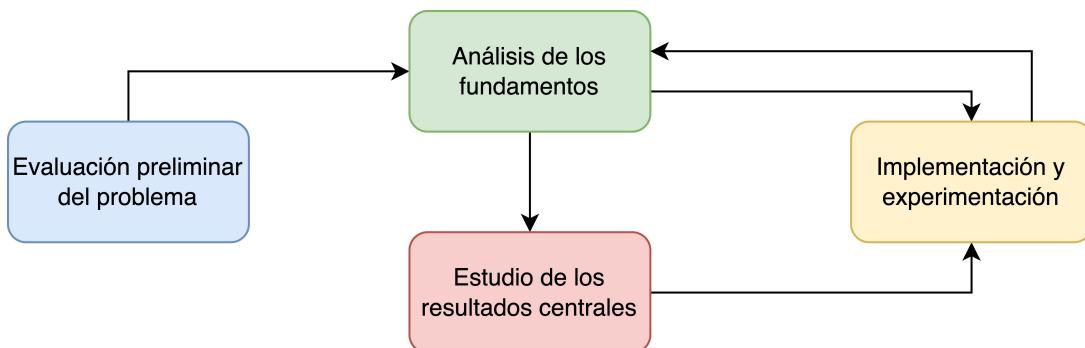


Figura 5.1: Diagrama del flujo del proyecto.

1. **Evaluación preliminar del problema:** Tras una primera revisión de la documentación en mayo y julio de 2023, se retomó la búsqueda y análisis del material en **enero y febrero** de 2024. Se identificaron los elementos necesarios para afrontar el estudio y se definió el alcance del trabajo. La intención primordial era conseguir digerir la demostración del [Teorema 3.3](#), de manera que se pudiese comprender de manera sencilla, ya que en el artículo original no se cuida tanto la exposición.
2. **Análisis de los fundamentos:** Una vez que ya se había acotado radio de acción, en **febrero y marzo** de 2023, se redactaron los fundamentos matemáticos básicos que se utilizan en la demostración de dicho teorema.

La pretensión era realizar una construcción de la noción de medida de Lebesgue, integral de Lebesgue y finalmente, la distancia en el espacio \mathcal{L}^1 de las funciones integrables, empezando desde el básico concepto de intervalo en \mathbb{R}^n y pasando por otros como el de función medible. Pese a ser conceptos que se desarrollan durante la carrera, y especialmente en la asignatura de Análisis II, se planteó esta parte de forma que pudiese servir de recurso útil a cualquier interesado en el tema, más allá de la relación con

las redes neuronales. Por ello, se vigiló con delicadeza la notación, las demostraciones y la coherencia global de la presentación. No se utiliza ningún componente que no haya sido demostrado, más allá de definiciones básicas de conceptos como compacidad, apertura, continuidad...

El proceder habitual es alcanzar la noción de integral de Lebesgue a través de las funciones simples. En nuestro caso, primero definimos la integral como la medida de la subgráfica de una función medible positiva, y más tarde hacemos las apreciaciones convenientes sobre funciones simples. Por ejemplo, esto permite prescindir del *Teorema de Aproximación de Lebesgue* en la presentación. También existen otras ventajas, como la facilidad para demostrar algunas de las propiedades de la integral, como el *Teorema de Convergencia Monótona* (véase la [Propiedad 1.6.7](#)), habiendo demostrado previamente algunas proposiciones sobre gráficas y las subgráficas. Sin embargo, también dificulta la prueba de otras, como la de la suma de la integral (véase la [Propiedad 1.6.6](#)).

Al fin y al cabo, la decisión de seguir este enfoque tiene que ver con la voluntad de poner en la práctica la destreza matemática alcanzada durante estos años. No busca ser una novedad, si bien es cierto que es complicado encontrar recursos que sigan este enfoque (algunos libros como [47] o [58] ofrecen algunas respuestas al respecto, pero no profundizan en exceso). Por tanto, muchas de los enunciados y demostraciones se han tenido que adaptar e incluso plantear desde cero, por no encontrarse referencias suficientes al respecto. Una de ellas es, por ejemplo, la de la [Proposición 1.2.1](#). Se propuso formalizar la prueba esta intuitiva propiedad de la medida elemental de los intervalos acotados, de la que apenas existe bibliografía, ya que se suele explicar en términos gráficos o necesita la introducción previa de varios resultados, que se salen demasiado del foco para un trabajo de estas características.

Hay que remarcar que, pese a que en algunas partes se sigue la estructura de las notas de clase [5], la organización está pensada para utilizar solo aquellos elementos que se van requiriendo para llegar a la distancia en \mathcal{L}^1 , más allá de algunas observaciones ilustrativas.

Se quiere poner en valor el trabajo realizado en esta parte ya que ha implicado la puesta en común de muchas ideas para poder proveer una exposición afinada y cohesionada.

3. **Estudio de los resultados centrales:** Una vez que se había reflexionado lo suficiente sobre la distancia en \mathcal{L}^1 y los conceptos previos, la demostración del [Teorema 3.3](#) resultó mucho más sencilla. Esta parte se realizó entre **marzo, abril y mayo** de 2024, e implicó la previa introducción de la formulación matemática de las redes neuronales que se expone en la [Sección 2.2](#), de nuevo siendo muy precisos para no dejar escapar ningún pormenor. Así, la demostración se pudo reformular con relativa facilidad para mayor claridad, realizándose además una serie de diagramas para visualizar las arquitecturas especificadas, mediante la herramienta *Draw.io* junto con *Adobe Photoshop*. Los diagramas siguen una configuración que se describe en la [Subsección 2.2.2](#).

El [Teorema 3.1](#) fue abordado después, ya con la mente puesta en la parte práctica. Su demostración implica conceptos que no se introdujeron previamente, por eso se centran los esfuerzos en proveer la intuición del mismo para funciones unidimensionales, que es lo que se hace durante toda la [Sección 3.1](#). Se complementa este desarrollo con otros resultados de índole similar.

Por último, una vez realizadas las reflexiones sobre estos dos teoremas, se procede a la explicación del resto de teoremas de nuestro artículo de referencia. En este caso no se

demuestran, solo se explican, y se relacionan entre ellos y con los anteriores teoremas. De esta forma, la [Subsección 3.2.4](#) supone una transición idónea hacia la parte práctica.

- 4. Implementación y experimentación:** En **mayo** de 2024, se comenzó a plantear la realización de un experimento que culminase satisfactoriamente el proyecto, para ver el comportamiento de algunos de los aspectos de las redes neuronales descritas en la práctica. Este experimento se acabó realizando entre los meses de **junio y julio**.

Se ideó un sistema para implementar las propias redes definidas mediante los teoremas de aproximación. Estas construcciones parecen poco prácticas a simple vista, puesto que si bien están acotadas en profundidad o anchura, el otro factor queda a merced del número de términos de una función simple que aproxima a la función original, que no tiene por qué ser manejable. Para salvar esta limitación, se planteó su implementación sobre la clase de las funciones lipschitzianas. Para ello, fue necesario volver al segundo paso e introducir algunos de los conceptos relativos. En este entorno, se pueden construir redes neuronales con un número de parámetros controlado que sean válidas para un análisis experimental. Una vez hecho esto, se puede entrenar estas arquitecturas mediante método de Descenso de Gradiente y técnicas relacionadas, que se habían introducido antes en la [Subsección 2.3.2](#) siguiendo algunas referencias como los apuntes de la asignatura de Visión por Computador [51]. Las conclusiones concretas del proceso se comentan exhaustivamente en la [Subsección 4.4.6](#). Con las herramientas que se conocían en profundidad y la gran cantidad de material estudiado, no fue complicado extraerlas.

En cuanto a la implementación, se realizó mediante un *notebook* en el lenguaje *Python*, ejecutado en un entorno de *Visual Studio Code* con el kernel de *Python 3.9.7*. Se utilizan librerías como *Tensorflow* (y módulos relacionados), *Pandas* y *Numpy*, entre otros, además de *Matplotlib* para las visualizaciones. Se puede encontrar en la plataforma *GitHub* [44]. Los requerimientos computacionales fueron escasos al tratarse de pruebas sencillas, por lo que se han ejecutado en una máquina estándar, sin necesidad de especificaciones muy avanzadas.

Como vemos, el esquema de diseño consta de unas fases muy diferenciadas, pero que han necesitado retroalimentarse a lo largo del proceso como fruto de la propia tarea de investigación. La **planificación temporal** comentada también se desglosa en la [Figura 5.2](#), ajustándose bastante a las expectativas iniciales.

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO
ETAPA 1							
ETAPA 2							
ETAPA 3							
ETAPA 4							

Figura 5.2: Tabla de planificación temporal del proyecto.

En cuanto a la **planificación económica**, tan solo se ha dispuesto de un ordenador estándar y una conexión a internet. No se ha necesitado utilizar servicios de ejecución en la nube. Se ha dedicado una media de 20 horas semanales al proyecto, y estableciendo un salario promedio de 20€ por hora, que se adecua aproximadamente a los estándares de un investigador en Inteligencia Artificial; con el trabajo total de unas 23 semanas, nos permite presupuestar el proyecto en alrededor de 9.200€, sin contabilizar la asistencia de los tutores.

5.2. Objetivos cumplidos

Durante la realización del trabajo, se superan con éxito los siguientes objetivos:

- Entendimiento de los **fundamentos matemáticos** relacionados con las funciones integramos en el sentido de Lebesgue, habiendo desarrollado algunas de las demostraciones de manera autónoma.
- Comprensión profunda del *Teorema de Cybenko* y el *Teorema de Aproximación Universal para Redes ReLU limitadas por anchura*, no solo a nivel técnico, sino a nivel visual. Con respecto a este último, se ha sido capaz de reformular la demostración presente en el apéndice de nuestro artículo base de manera que sea más clarividente, puliendo la notación así como algunos detalles ambiguos, innecesarios o incluso erróneos.
- Análisis de la **pérdida de expresividad** de las redes neuronales ReLU acotadas en anchura, relacionando diferentes puntos de vista, con especial hincapié en el aspecto de su eficiencia con respecto de la profundidad.
- Desarrollo de una implementación para construir automáticamente **redes neuronales explícitas** que aproximen funciones a partir de unos datos, sin necesidad de un proceso iterativo de entrenamiento, y siendo precisos con los criterios específicos de aplicación.
- Análisis de diversas arquitecturas neuronales, con **distintas disposiciones de sus neuronas**, y su comportamiento en la práctica cuando se les entrena con métodos convencionales, observando cuidadosamente el tipo de ajuste que realizan y su capacidad de adaptación a datos no vistos con anterioridad. Se compara además el efecto de la utilización de pesos iniciales predeterminados y pesos iniciales aleatorios, y sus repercusiones en la convergencia en el aprendizaje.

A nivel personal, la realización del trabajo me ha resultado muy satisfactoria. He podido desplegar con éxito muchas de las habilidades que he ido adquiriendo durante la carrera, entremezclando de manera armoniosa las disciplinas de Informática y Matemáticas.

He afianzado bastantes conceptos, descubierto muchos otros y me he tomado el tiempo de tratarlos con el máximo rigor posible. Además, este desarrollo me ha suscitado un interés genuino en la investigación, que seguiré aplicando en mi posterior formación académica y profesional.

5.3. Trabajo futuro

Las siguientes tareas se plantean como continuación del presente trabajo:

- Desarrollo de un algoritmo eficiente para la determinación de una función simple que aproxime en \mathcal{L}^1 funciones lipschitzianas definidas en \mathbb{R}^n , para cualquier $n \in \mathbb{N}$. La construcción de este algoritmo permitiría extender de manera sencilla todo el desarrollo del [Capítulo 4](#) a contextos más generales.

Además, el problema de determinar cuál es la función simple con el menor número de términos que aproxima una función lipschitziana dado un conjunto restringido de datos muestrados de ella es *NP-difícil*, ya que se encuentra en la línea de problemas como el descrito en [29]. Su estudio exhaustivo puede contribuir significativamente a la rama de la computación teórica.

- Planteamiento de métodos que favorezcan el entrenamiento de redes profundas y estrechas (con pocas neuronas por capa), con especial atención a la inicialización de los pesos y biases, como se intenta en trabajos como [31].
- Estudio del efecto de la reducción en anchura y/o profundidad de las arquitecturas neuronales en el contexto de los problemas reales. Una forma de hacerlo sería entrenar una red base para uno (o varios) conjuntos de datos y transferir el conocimiento hacia redes con un número parecido de parámetros pero distinta disposición, mediante el modelo maestro-estudiante propio de la estrategia de Destilación del Conocimiento (*Knowledge Distillation*) [21, 61]. Con un enfoque adecuado, como por ejemplo el basado en características (*Feature Based*) [23], podríamos entender de qué forma aprende cada red estudiante y cuáles son sus limitaciones para replicar el comportamiento de la red maestro, según el caso. En artículos como [7] se aplican ideas similares.

Este estudio se puede llevar a cabo directamente sobre Redes Neuronales Totalmente Conectadas, o ampliarlo a redes adaptadas a la resolución de otro tipo de problemas, como las Redes Neuronales Convolucionales (CNNs) [45]. Existen una gran diversidad de arquitecturas de este tipo cuyas últimas capas son totalmente conectadas (véase ResNet [22], VGG16 [53], etc).

- Extensión de los argumentos teóricos de expresividad al contexto de otras funciones de activación que se suelen emplear, como Leaky ReLU. Así mismo, se podría investigar la inclusión de funciones de activación distintas en las capas ocultas, o incluso dentro de las mismas capas.

De esta manera, se abre la posibilidad de continuar con nuevos caminos de investigación y, poco a poco, descifrar los entresijos del funcionamiento de las redes neuronales. Esto permitirá fundamentar las bases y contribuir a avances punteros que impulsen estas tecnologías a un nivel (incluso) superior.

Bibliografía

- [1] Build your own neural network classifier in R — junma5.weebly.com. <https://junma5.weebly.com/data-blog/build-your-own-neural-network-classifier-in-r>.
- [2] Underfitting and Overfitting in Machine Learning | Baeldung on Computer Science — baeldung.com. <https://www.baeldung.com/cs/ml-underfitting-overfitting>, 2024.
- [3] Grant Sanderson (3Blue1Brown). Neural networks, 2024. https://youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&si=0gOCscYTU2XLssby.
- [4] Ali Azawii Abdul lateef, Sufyan T. Faraj Al-Janabi, and Belal Al-Khateeb. Survey on intrusion detection systems based on deep learning. *Periodicals of Engineering and Natural Sciences (PEN)*, 7(3):1074, August 2019.
- [5] Rafael Payá Albert. *Notas de Análisis II*. Universidad de Granada, Granada.
- [6] Motasem Alfarra, Adel Bibi, Hasan Hammoud, Mohamed Gaafar, and Bernard Ghanem. On the decision boundaries of deep neural networks: A tropical geometry perspective. *CoRR*, abs/2002.08838, 2020.
- [7] Takanori Ashihara, Takafumi Moriya, Kohei Matsuura, and Tomohiro Tanaka. Deep versus wide: An analysis of student architectures for task-agnostic knowledge distillation of self-supervised speech models, 2022.
- [8] Ashyi. Exponentially Weighted Average for Deep Neural Networks — medium.datadriveninvestor.com. <https://medium.datadriveninvestor.com/exponentially-weighted-average-for-deep-neural-networks-39873b8230e9>.
- [9] Chaity Banerjee, Tathagata Mukherjee, and Eduardo Pasiliao. Feature representations using the reflected rectified linear unit (rrelu) activation. *Big Data Mining and Analytics*, 3(2):102–120, 2020.
- [10] Ilya Baran. Adaptive algorithms for problems involving blackbox lipschitz functions. 06 2005.
- [11] Andrew Barron. Approximation and estimation bounds for artificial neural networks. *Machine Learning*, 14:115–133, 01 1994.
- [12] Daniel Berrar. *Cross-Validation*, page 542–545. Elsevier, 2019.
- [13] G. Cyberko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, December 1989.
- [14] Saeed Damadi, Golnaz Moharrer, and Mostafa Cham. The backpropagation algorithm for a math student, 2023.
- [15] Amit Daniely. Depth separation for neural networks. *CoRR*, abs/1702.08489, 2017.
- [16] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- [17] Synho Do, Kyoung Doo Song, and Joo Won Chung. Basics of deep learning: A radiologist’s guide to understanding published radiology articles on deep learning. *Korean Journal of Radiology*, 21(1):33, 2020.
- [18] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. *CoRR*, abs/1512.03965, 2015.
- [19] G.B. Folland. *Real Analysis: Modern Techniques and Their Applications*. Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts. Wiley, 2013.

Bibliografía

- [20] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.
- [21] Jianping Gou, Baosheng Yu, Stephen John Maybank, and Dacheng Tao. Knowledge distillation: A survey. *CoRR*, abs/2006.05525, 2020.
- [22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [23] Hyeonseok Hong and Hyun Kim. Feature distribution-based knowledge distillation for deep neural networks. In *2022 19th International SoC Design Conference (ISOCC)*, pages 75–76, 2022.
- [24] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, jul 1989.
- [25] Christopher Williams John McGonagle, George Shaikouski. Backpropagation. July 8th, 2024.
- [26] Nitish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tang. On large-batch training for deep learning: Generalization gap and sharp minima. 09 2016.
- [27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.
- [28] Nalinda Kulathunga, Nishath Rajiv Ranasinghe, Daniel Vrinceanu, Zackary Kinsman, Lei Huang, and Yunjiao Wang. Effects of nonlinearity and network architecture on the performance of supervised neural networks. *Algorithms*, 14(2):51, February 2021.
- [29] Aline A.S. Leao, Franklina M.B. Toledo, José Fernando Oliveira, Maria Antónia Carraville, and Ramón Alvarez-Valdés. Irregular packing problems: A review of mathematical models. *European Journal of Operational Research*, 282(3):803–822, May 2020.
- [30] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.
- [31] Hyunwoo Lee, Yunho Kim, Seung Yeop Yang, and Hayoung Choi. Improved weight initialization for deep and narrow feedforward neural network. *Neural Networks*, 176:106362, August 2024.
- [32] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.
- [33] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J. Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’14, page 661–670, New York, NY, USA, 2014. Association for Computing Machinery.
- [34] Octavio Loyola-González. Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view. *IEEE Access*, 7:154096–154113, 2019.
- [35] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples. 2019.
- [36] Lu Lu, Yanhui Su, and George Em Karniadakis. Collapse of deep and narrow neural nets. 2018.
- [37] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width, 2017.
- [38] Stamatis Mastromichalakis. Alrelu: A different approach on leaky relu activation function to improve neural networks performance. *CoRR*, abs/2012.07564, 2020.
- [39] Lawrence Narici. The hahn-banach theorem. *Advanced Courses of Mathematical Analysis II*, 01 2007.
- [40] Meenal V. Narkhede, Prashant P. Bartakke, and Mukul S. Sutaone. A review on weight initialization strategies for neural networks. *Artificial Intelligence Review*, 55(1):291–322, June 2021.
- [41] Graham Neubig. Neural machine translation and sequence-to-sequence models: A tutorial, 2017.

- [42] Andrew Y. Ng. Feature selection, l₁ vs. l₂ regularization, and rotational invariance. In *Twenty-first international conference on Machine learning - ICML '04*, ICML '04. ACM Press, 2004.
- [43] Michael A. Nielsen. Neural networks and deep learning, 2018.
- [44] Luis Crespo Orti. tfg-analisis-eficiencia-anchura. <https://github.com/luiscrespo/TFG-analisis-eficiencia-anchura.git>, 2024.
- [45] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.
- [46] Marco Pavone. On the riesz representation theorem for bounded linear functionals. *Mathematical Proceedings of the Royal Irish Academy*, 94A:133–135, 01 1994.
- [47] Charles C. Pugh. *Real Mathematical Analysis*. Springer International Publishing, 2015.
- [48] H. L. Royden. *Real analysis [by] H. L. Royden*. Macmillan, New York, 2d ed. edition, 1968.
- [49] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.
- [50] Itay Safran and Ohad Shamir. Depth separation in relu networks for approximating smooth non-linear functions. *ArXiv*, abs/1610.09887, 2016.
- [51] Pablo Mesejo Santiago. *Notas de Visión por Computador*. Universidad de Granada, Granada.
- [52] Valery Serov. *Fourier Series, Fourier Transform and Their Applications to Mathematical Physics*. 01 2017.
- [53] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.
- [54] Himanshi Singh. Variants of gradient descent algorithm, 2023. 07-08-2024.
- [55] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, jan 2014.
- [56] Elias M Stein and Rami Shakarchi. *Real analysis: measure theory, integration, and Hilbert spaces*. Princeton lectures in analysis. Princeton Univ. Press, Princeton, NJ, 2005.
- [57] Tomasz Szandała. Review and comparison of commonly used activation functions for deep neural networks. 2020.
- [58] Terence Tao. *An Introduction to Measure Theory*. American Mathematical Society, 2011.
- [59] Matus Telgarsky. Benefits of depth in neural networks. *CoRR*, abs/1602.04485, 2016.
- [60] Gal Vardi, Gilad Yehudai, and Ohad Shamir. Width is less important than depth in relu neural networks. *CoRR*, abs/2202.03841, 2022.
- [61] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *CoRR*, abs/2004.05937, 2020.
- [62] Halbert White. A reality check for data snooping. *Econometrica*, 68(5):1097–1126, September 2000.
- [63] D.Randall Wilson and Tony R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451, December 2003.
- [64] Jin Xu, Zishan Li, Bowen Du, Miaomiao Zhang, and Jing Liu. Reluplex made more practical: Leaky relu. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, 2020.
- [65] Denny Zhou, Mao Ye, Chen Chen, Tianjian Meng, Mingxing Tan, Xiaodan Song, Quoc Le, Qiang Liu, and Dale Schuurmans. Go wide, then narrow: Efficient training of deep thin networks, 2020.