



# UNIVERSIDAD DE GRANADA

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

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y  
MATEMÁTICAS

TRABAJO DE FIN DE GRADO

## Análisis teórico y empírico del Deep Double Descent

Presentado por:  
Juan Antonio Ruiz Arévalo

Tutorizado por:  
Francisco Javier Merí de la Maza  
Pablo Mesejo Santiago

Curso académico 2024-2025

# Análisis teórico y empírico del Deep Double Descent

Juan Antonio Ruiz Arévalo

Juan Antonio Ruiz Arévalo *Análisis teórico y empírico del Deep Double Descent.*  
Trabajo de fin de Grado. Curso académico 2024-2025.

<b>Responsable de tutorización</b>	Francisco Javier Merí de la Maza <i>Departamento de Análisis Matemático</i>	Doble Grado en Ingeniería Informática y Matemáticas
	Pablo Mesejo Santiago <i>Departamento de Ciencias de la Computación e Inteligencia Artificial</i>	Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación y Facultad de Ciencias

Universidad de Granada



**DECLARACIÓN DE ORIGINALIDAD**

D. Juan Antonio Ruiz Arévalo

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

En Granada a 3 de junio de 2025

Fdo: Juan Antonio Ruiz Arévalo



---

**D. Francisco Javier Merí de la Maza**, Profesor Titular de Universidad del Departamento de **Análisis Matemático** de la Universidad de Granada.

**D. Pablo Mesejo Santiago**, Profesor Contratado Doctor Indefinido del Departamento de **Ciencias de la Computación e Inteligencia Artificial** de la Universidad de Granada.

**Informan:**

Que el presente trabajo, titulado *Análisis teórico y empírico del Deep Double Descent*, ha sido realizado bajo su supervisión por **Juan Antonio Ruiz Arévalo**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 3 de junio de 2025.

**Los directores:**

Francisco Javier Merí de la Maza

Pablo Mesejo Santiago

*A mis padres, a mi hermana  
y a toda mi familia.*

## **Agradecimientos**

*En primer lugar, quiero expresar mi más sincero agradecimiento a mis tutores, Javier y Pablo, por brindarme la posibilidad de desarrollar este proyecto con vosotros y por darme total libertad en su desarrollo. Agradezco profundamente vuestra disponibilidad para resolver cualquier duda, incluso antes de comenzar con el trabajo, así como vuestra motivación e incansable ayuda en todo lo que he necesitado.*

*Extender mi gratitud a la Universidad de Granada por darme la oportunidad de continuar mis estudios en tan prestigiosa institución, cuyo entorno académico y humano ha sido clave en mi desarrollo personal y profesional.*

*Agradecer también a todos los compañeros y amigos que he tenido el privilegio de conocer a lo largo de mi formación. En especial, a Julio y Guillermo, por estar a mi lado en los momentos más complicados y por celebrar cada logro de forma conjunta. Nada de esto hubiera sido lo mismo sin vuestra ayuda y compañía.*

*Finalmente, y lo más importante de todo, quiero agradecer a toda mi familia, en especial a mis padres y a mi hermana, por apoyarme de manera incondicional desde la distancia para llegar hasta aquí. Vuestro cariño y respaldo han sido fundamentales durante todo el camino.*



## Resumen

**Palabras clave:** Aprendizaje Automático, Aprendizaje Profundo, Inteligencia Artificial, Equilibrio Sesgo-Varianza, Aproximación No Lineal, Marcos de Generalización, Explicabilidad

El aprendizaje automático y, por extensión, el aprendizaje profundo, se ha convertido en una herramienta fundamental en numerosos ámbitos, desempeñando un papel determinante al complementar y dar apoyo al experto humano en numerosas tareas. No obstante, el entrenamiento de modelos cada vez más complejos ha revelado, recientemente, comportamientos inesperados en su rendimiento, entre los cuales destaca el fenómeno conocido como *Deep Double Descent*. Esto desafía la sabiduría clásica del aprendizaje, al evidenciar que la relación entre la complejidad del modelo y su rendimiento no se ajusta a las curvas previstas por la teoría convencional.

Los conceptos clásicos del aprendizaje resultan insuficientes para explicar ciertos fenómenos emergentes. De hecho, algunos, como el equilibrio entre sesgo y varianza, parecen contradecir estas nuevas tendencias. Este suceso revela una brecha significativa entre los conocimientos teóricos y los resultados empíricos observados en la actualidad, lo cual nos recuerda que el campo de la inteligencia artificial está en constante evolución, y que los avances en esta disciplina no se realizan a la misma velocidad a nivel teórico y práctico.

En este TFG nos centraremos en exponer los principios informáticos y matemáticos subyacentes necesarios para comprender el *Deep Double Descent*, partiendo del marco clásico del equilibrio entre sesgo y varianza, para lo cual será necesario recurrir a nociones de probabilidad y estadística. Asimismo, se revisarán e introducirán conceptos relativos al álgebra matricial que resultan imprescindibles para un entendimiento más profundo. Sin embargo, dada la percepción común de las redes neuronales como «cajas negras», atribuida a la dificultad de interpretar su funcionamiento interno, en este trabajo se pretende analizar y contribuir a una mejor comprensión del comportamiento general de redes neuronales profundas.

Este estudio se centra en la definición de nuevos marcos de generalización, conocidos como sesgos inductivos (*inductive bias*), que proporcionen explicabilidad más allá de lo que ofrece la teoría clásica, unificando así los conceptos tradicionales con los enfoques modernos. Además, se recurre a la aproximación no lineal como vía complementaria para reforzar dichos marcos desde una perspectiva más matemática y formal.

A través de diversos experimentos, tanto en arquitecturas simples como en arquitecturas convolucionales avanzadas, como ResNet, se demuestra que el *Deep Double Descent* no solo mejora el rendimiento de los modelos en tareas de clasificación y regresión, sino que también sugiere que las soluciones obtenidas tienden a ser cada vez más simples, siguiendo la filosofía del principio de la navaja de Ockham.

En conclusión, este trabajo no se centra únicamente en ofrecer una explicación del *Deep Double Descent*, sino que también constituye un avance hacia la comprensión y explicabilidad de las nuevas tendencias de generalización. Este comportamiento apunta hacia una nueva dirección en el aprendizaje automático, basada en la idea de «*cuanto más grande, mejor*», sin olvidar que esto conlleva importantes desafíos de eficiencia computacional y escalabilidad, lo que sigue subrayando la necesidad de equilibrar ambición tecnológica y viabilidad práctica.

## Summary

**Keywords:** Machine Learning, Deep Learning, Artificial Intelligence, Bias-Variance Tradeoff, Non-Linear Approximation, Generalization Frameworks, Explainability

Machine learning and, by extension, deep learning, has become a fundamental tool in many fields, playing a decisive role in complementing and supporting the human expert in numerous tasks. However, the training of increasingly complex models has recently revealed unexpected behaviors in their performance, among which the phenomenon known as *Deep Double Descent* stands out. This behavior challenges the classical wisdom of learning by showing that the relationship between model complexity and performance does not conform to the learning curves predicted by conventional theory.

The classical concepts of learning are insufficient to explain certain emerging phenomena. In fact, some of them, such as the bias-variance trade-off, seem to contradict these new trends. This event reveals a significant gap between theoretical knowledge and the empirical findings recorded in recent observations, which reminds us that the field of AI is constantly evolving, and that advances in this discipline do not occur at the same pace on the theoretical and practical levels.

In this Final Degree Project, we will focus on exposing the underlying computational and mathematical principles necessary to understand the *Deep Double Descent*, starting from the classical framework of the bias-variance trade-off, for which it will be necessary to draw on notions of probability and statistics. Likewise, concepts related to matrix algebra, which are essential for a deeper understanding, will be reviewed and introduced. However, given the common perception of neural networks as “black boxes”, attributed to the difficulty of interpreting their inner workings, this paper aims to analyze and contribute to a better understanding of the general behavior of deep neural networks.

This study focuses on the definition of new generalization frameworks, known as inductive biases, that provide explanability beyond what is offered by classical theory, thus unifying traditional concepts with modern approaches. In addition, the nonlinear approach is used as a complementary way to reinforce these frameworks from a more mathematical and formal perspective.

Through various experiments, both in simple architectures and in advanced convolutional architectures, such as ResNet, it is shown that the *Deep Double Descent* not only improves the performance of the models in classification and regression tasks, but also suggests that the solutions obtained tend to be increasingly simpler, following the philosophy of Ockham’s razor principle.

In conclusion, this work is not only focused on providing an explanation of the *Deep Double Descent*, but also constitutes an advance towards the understanding and explanability of new generalization trends. This behavior points towards a new direction in machine learning, based on the idea of «*the bigger, the better*», without forgetting that this brings with it significant computational efficiency and scalability challenges, which continues to underscore the need to balance technological ambition and practical feasibility.



# Índice general

<b>Agradecimientos</b>	<b>V</b>
<b>Resumen</b>	<b>VII</b>
<b>Summary</b>	<b>VIII</b>
<b>Índice de figuras</b>	<b>XIII</b>
<b>Índice de tablas</b>	<b>XV</b>
<b>Introducción</b>	<b>XVII</b>
1. Definición del problema . . . . .	XVIII
2. Motivación . . . . .	XIX
3. Objetivos . . . . .	XX
3.1. Objetivo matemático . . . . .	XX
3.2. Objetivo informático . . . . .	XXI
4. Planificación del proyecto . . . . .	XXI
<b>I. Fundamentos Teóricos</b>	<b>1</b>
<b>1. Probabilidad</b>	<b>3</b>
1.1. Espacios de probabilidad y $\sigma$ -álgebras . . . . .	3
1.2. Variables aleatorias y esperanza . . . . .	4
1.2.1. Probabilidad condicional . . . . .	6
1.2.2. Independencia de variables aleatorias . . . . .	7
1.2.3. Propiedades de la esperanza y varianza . . . . .	10
1.3. Distribuciones de probabilidad . . . . .	12
1.3.1. Distribución Normal . . . . .	12
1.3.2. Distribución de Rademacher . . . . .	14
<b>2. Descomposición en Valores Singulares y Pseudoinversa de una Matriz</b>	<b>15</b>
2.1. Vectores y matrices . . . . .	15
2.2. SVD y pseudoinversa . . . . .	17
<b>3. Aprendizaje Automático y Aprendizaje Profundo</b>	<b>21</b>
3.1. Fundamentos . . . . .	21
3.2. Redes neuronales artificiales . . . . .	22
3.2.1. Redes neuronales convolucionales . . . . .	23
<b>4. El Dilema Clásico del Aprendizaje</b>	<b>28</b>
4.1. Concepto de aprendizaje . . . . .	28
4.1.1. Descenso de gradiente y aprendizaje . . . . .	29

4.2.	<i>Bias-variance tradeoff</i> . . . . .	33
4.2.1.	Formulación matemática del $E_{out}$ . . . . .	33
4.3.	Equilibrio clásico entre sesgo y varianza . . . . .	39
4.3.1.	Curva de aprendizaje . . . . .	40
4.4.	<i>Underfitting</i> y <i>overfitting</i> . . . . .	42
4.5.	Marcos de generalización clásicos . . . . .	44
<b>II.</b>	<b>Estado del Arte</b>	<b>49</b>
<b>5.</b>	<b>Trabajos Relacionados</b>	<b>51</b>
5.1.	Origen y primeras manifestaciones . . . . .	52
5.2.	El nacimiento del <i>Deep Double Descent</i> . . . . .	53
5.3.	Avances recientes . . . . .	54
<b>III.</b>	<b>Análisis Teórico y Empírico</b>	<b>57</b>
<b>6.</b>	<b>Análisis Teórico del Deep Double Descent</b>	<b>59</b>
6.1.	Planteamiento teórico . . . . .	59
6.1.1.	Análisis intuitivo en un problema de mínimos cuadrados . . . . .	62
6.2.	Convergencia del descenso de gradiente . . . . .	65
6.2.1.	Problema de regresión . . . . .	66
6.2.2.	Problema de clasificación con datos separables . . . . .	69
6.3.	Optimización en la zona sobreparametrizada . . . . .	71
6.3.1.	El impacto del sesgo inductivo en la selección de hipótesis . . . . .	76
6.3.2.	PAC-Bayes y límite de hipótesis numerables . . . . .	80
6.4.	Aproximación no lineal . . . . .	81
6.4.1.	Aproximación en un espacio de Hilbert . . . . .	82
6.4.2.	Aproximación altamente no lineal . . . . .	84
6.4.3.	Analogía con el <i>Deep Double Descent</i> . . . . .	88
6.5.	Discusión final . . . . .	89
<b>7.</b>	<b>Análisis Empírico del Deep Double Descent</b>	<b>91</b>
7.1.	Materiales y métodos . . . . .	91
7.1.1.	<i>Datasets</i> . . . . .	91
7.1.2.	Protocolo de validación experimental . . . . .	92
7.1.3.	Arquitecturas utilizadas . . . . .	93
7.1.4.	Hiperparámetros . . . . .	96
7.2.	Experimentos . . . . .	97
7.2.1.	Entornos de desarrollo y ejecución . . . . .	97
7.2.2.	Aproximación polinómica . . . . .	99
7.2.3.	<i>Noise-wise double descent</i> . . . . .	105
7.2.4.	<i>Sample-wise double descent</i> . . . . .	107
7.2.5.	<i>Model &amp; Epoch-wise double descent</i> . . . . .	110
7.2.6.	<i>Width vs Depth double descent</i> . . . . .	116
7.3.	Discusión comparativa global . . . . .	117

*Índice general*

<b>IV. Conclusiones y Trabajos Futuros</b>	<b>119</b>
8. Conclusiones	121
9. Trabajos futuros	123
<b>V. Anexos</b>	<b>124</b>
A. Detalles Matemáticos Adicionales	126
B. Experimentos Adicionales: Hiperparámetros	129
C. Descenso de Gradiente en un Problema OLS	131
D. Irregularidades en la Dinámica del Error	132
Glosario matemático	135
Glosario informático	137
Bibliografía	139

# Índice de figuras

1.	Ejemplo de <i>Deep Double Descent</i> en ResNet18 [NKB <sup>+</sup> 19]. . . . .	xviii
2.	Modelo en cascada realimentado. . . . .	xxii
1.1.	Comparación de distribuciones unimodal y multimodal. . . . .	13
1.2.	Ejemplos de distribuciones normales. . . . .	14
3.1.	Ejemplos de problemas de clasificación y regresión. . . . .	21
3.2.	Ejemplos de neurona biológica [NGLK18] y neurona artificial (basada en [LJY24]). . . . .	22
3.3.	Ejemplo de CNN utilizada para clasificación de imágenes [Swa20]. . . . .	24
3.4.	Ejemplo de convolución con <i>padding</i> [Sah18]. . . . .	25
3.5.	Ejemplos de <i>pooling</i> utilizados en CNN. . . . .	26
3.6.	Ejemplos de funciones de activación utilizadas en CNN. . . . .	27
4.1.	Diagrama representando el concepto clásico de aprendizaje. . . . .	29
4.2.	Distintas tasas de aprendizaje para el descenso de gradiente [AMMIL12]. . . . .	30
4.3.	Proceso de retropropagación del error [Biso6]. . . . .	32
4.4.	Distintos casos del conjunto de hipótesis y de la función objetivo. . . . .	40
4.5.	Ejemplo de curva de aprendizaje tradicional [AMMIL12]. . . . .	41
4.6.	Ejemplos de curvas de aprendizaje modificadas para este proyecto. . . . .	41
4.7.	Relación <i>bias/variance</i> con <i>underfitting</i> y <i>overfitting</i> en el contexto clásico. . . . .	44
4.8.	Error en función de la dimensión VC [AMMIL12]. . . . .	47
5.1.	Número de publicaciones relativas al <i>Deep Double Descent</i> en función del año de publicación. . . . .	51
5.2.	<i>Deep Double Descent</i> presente en ADALINE. . . . .	52
5.3.	Curva del error unificada entre la teoría clásica y moderna [BHMM19]. . . . .	54
5.4.	Groking y <i>Deep Double Descent</i> . . . . .	55
6.1.	Ejemplo de doble descenso con las distintas zonas de parametrización. . . . .	60
6.2.	Curvas típicas del error en función del sesgo y la varianza [YYY <sup>+</sup> 20]. . . . .	61
6.3.	Ejemplos de distribuciones de Marchenko-Pastur [MM18]. . . . .	65
6.4.	Ejemplos de paisajes de la función de pérdida en redes neuronales [LZB21]. . . . .	72
6.5.	Ejemplo de función de pérdida que satisface la condición $\mu$ -PL [LZB21]. . . . .	76
6.6.	Distintos modelos polinómicos para aproximar una función, inspirada en [SKR <sup>+</sup> 23]. . . . .	77
6.7.	Fronteras de decisión para modelos con distinta capacidad [SFB <sup>+</sup> 22]. . . . .	78
6.8.	Diferentes tipos de sesgo para el espacio de hipótesis [Wil25]. . . . .	79
6.9.	Aumentar la capacidad de un modelo mejora su generalización [Wil25]. . . . .	80
7.1.	Protocolos de validación experimental. . . . .	93
7.2.	Arquitectura 2NN. . . . .	94
7.3.	Arquitectura 3CNN. . . . .	95
7.4.	Arquitectura ResNet18 modificada. . . . .	96

7.5.	Doble descenso al utilizar aproximación polinómica de Legendre y norma del vector de parámetros. . . . .	101
7.6.	Intuición del <i>Deep Double Descent</i> usando regresión polinómica. . . . .	102
7.7.	Error en entrenamiento y test para las distintas aproximaciones polinómicas. .	102
7.8.	Normas del vector de parámetros para las aproximaciones polinómicas clásicas. .	103
7.9.	Aproximaciones polinómicas clásicas de la función objetivo. . . . .	103
7.10.	Aproximaciones polinómicas de Legendre de la función objetivo, tanto sin ruido como con ruido. . . . .	104
7.11.	Ejemplos de doble descenso en regresión polinómica de Legendre para una función objetivo, tanto con ruido como sin ruido. . . . .	105
7.12.	<i>Deep Double Descent</i> para distintos niveles de ruido. . . . .	106
7.13.	Umbral de interpolación para el doble descenso con distintos niveles de ruido. .	106
7.14.	Ejemplo de <i>sample-wise double descent</i> . . . . .	108
7.15.	Ratio parámetros frente a número de ejemplos en el doble descenso. . . . .	109
7.16.	Doble descenso en función del tamaño del modelo y del número de épocas para la red 2NN y un subconjunto de MNIST. . . . .	110
7.17.	Error en entrenamiento y test en función del tamaño del modelo y del número de épocas para la red 2NN y un subconjunto de CIFAR10. . . . .	111
7.18.	Doble descenso en función del tamaño del modelo y del número de épocas para la red 3CNN y un subconjunto de MNIST. . . . .	112
7.19.	Doble descenso en función del tamaño del modelo y del número de épocas para la red 3CNN y un subconjunto de CIFAR10. . . . .	113
7.20.	Doble descenso en función del tamaño del modelo y del número de épocas para la red 3CNN y un subconjunto de CIFAR100. . . . .	113
7.21.	Doble descenso en función del tamaño del modelo y del número de épocas para la red ResNet18 y el conjunto MNIST. . . . .	114
7.22.	Doble descenso en función del tamaño del modelo y del número de épocas para la red ResNet18 y el conjunto CIFAR10. . . . .	114
7.23.	Doble descenso en función del tamaño del modelo y del número de épocas para la red ResNet18 y el conjunto CIFAR100. . . . .	115
7.24.	Doble descenso en función del número de épocas para dos arquitecturas ResNet18 sobreparametrizadas y el conjunto CIFAR10. . . . .	116
7.25.	Comparativa del doble descenso entre arquitecturas anchas y profundas. . . . .	117
B.1.	<i>Deep Double Descent</i> para distintos tamaños de <i>batch</i> . . . . .	129
B.2.	<i>Deep Double Descent</i> para distinto <i>learning rate</i> . . . . .	130
C.1.	Error en entrenamiento y test para las bases de Legendre y clásica utilizando el GD. . . . .	131
D.1.	Comparativa <i>epoch-wise double descent</i> entre un modelo infraparametrizado y uno sobreparametrizado en el que aparecen picos significativos en el error. .	132
D.2.	Error en test respecto a distintas configuraciones altamente sobreparametrizadas de la red 2NN. . . . .	133
D.3.	Paisajes de la función de pérdida para distintos modelos sobreparametrizados [LXT <sup>+</sup> 18]. . . . .	134

# Índice de tablas

1.	Planificación temporal inicial del proyecto. . . . .	xxii
2.	Planificación temporal final del proyecto. . . . .	xxiii
3.	Resumen de las fechas de inicio y fin del proyecto, junto con la duración total. . . . .	xxiv
4.	Estimación del coste del proyecto. . . . .	xxiv
5.1.	Resumen de los principales artículos junto con sus contribuciones. . . . .	56
6.1.	Resumen comparativo de las diferencias entre el paradigma clásico y el paradigma moderno. . . . .	90
7.1.	Resumen de los <i>datasets</i> utilizados. . . . .	92
7.2.	Resumen de las arquitecturas 2NN. . . . .	94
7.3.	Número de parámetros de las arquitecturas 3CNN. . . . .	95
7.4.	Número de parámetros de las arquitecturas ResNet18 modificadas. . . . .	96
7.5.	Resumen de las ideas principales de los experimentos realizados. . . . .	98
7.6.	Resumen del número de experimentos junto con el número de horas totales de GPU. . . . .	99
7.7.	Aproximaciones polinómicas de grado $n$ utilizadas para regresión polinomial. . . . .	100
7.8.	Resumen de los experimentos para el DDD por nivel de ruido. . . . .	106
7.9.	Resumen de los experimentos para el doble descenso por número de ejemplos de entrenamiento. . . . .	108
7.10.	Resumen del ratio parámetros/ejemplos. . . . .	109
7.11.	Resumen de los experimentos para el doble descenso por complejidad del modelo y épocas para la red 2NN. . . . .	111
7.12.	Resumen de los experimentos para el doble descenso por complejidad del modelo y épocas para la arquitectura 3CNN. . . . .	112
7.13.	Resumen de los experimentos para el doble descenso por complejidad del modelo y épocas para la red ResNet18 modificada. . . . .	115
7.14.	Resumen de los experimentos realizados sobre el <i>Deep Double Descent</i> . . . . .	118
B.1.	Resumen de los experimentos realizados para el <i>Deep Double Descent</i> con distinto tamaño de <i>batch</i> . . . . .	130
B.2.	Resumen de los experimentos realizados para el DDD con distinto <i>learning rate</i> . . . . .	130



# Introducción

En el contexto de la actual revolución tecnológica, diversas arquitecturas de redes neuronales han demostrado ser especialmente eficaces en tareas complejas del ámbito del aprendizaje profundo. Modelos como las redes convolucionales, los Transformers y los modelos de difusión han mostrado un rendimiento notable en campos como la visión por computador, el procesamiento del lenguaje natural y la generación de contenido, debido a su gran capacidad para analizar grandes volúmenes de datos. En este trabajo, nos centraremos en las redes neuronales convolucionales, dada su consolidada capacidad para extraer representaciones significativas de la información visual, logrando en muchos casos superar el rendimiento alcanzado por los humanos en tareas de clasificación y detección de imágenes.

En los últimos años y bajo este marco de expansión y evolución del aprendizaje, el *Deep Double Descent (DDD)* ha surgido como un importante campo de interés dentro de este ámbito, desafiando los principios clásicos del aprendizaje estadístico. La sabiduría tradicional sugiere que, a medida que aumenta la complejidad del modelo, el error fuera de la muestra disminuye hasta alcanzar un mínimo para, a continuación, aumentar debido al sobreajuste (formando la tradicional curva con forma de "U").

Este concepto, conocido como equilibrio entre sesgo y varianza, difiere de las recientes observaciones obtenidas, especialmente en modelos de aprendizaje profundo, en los que puede producirse una segunda disminución del error fuera de la muestra, alcanzando un nuevo mínimo y formando una nueva gráfica del error de generalización que presenta dos descensos. Este novedoso hecho pone en tela de juicio la sabiduría clásica sobre el tema y proporciona nuevas perspectivas a la hora de crear y entrenar los modelos.

Como consecuencia, este Trabajo de Fin de Grado (TFG) se ocupa de explorar el concepto de *Deep Double Descent*, sus fundamentos teóricos y sus implicaciones para el aprendizaje automático moderno. Un área clave de interés es cómo este fenómeno se manifiesta en redes neuronales profundas, conocidas por su enorme complejidad en cuanto a número de parámetros se refiere y su potencial de sobreajuste. A pesar de que los modelos de aprendizaje profundo han tenido gran éxito en diversas aplicaciones, como la visión por computador o el procesamiento de lenguaje natural, la curva del doble descenso aporta nuevo conocimiento sobre cómo estos modelos pueden llegar a generalizar en un régimen que ha sido vagamente estudiado y explorado: el régimen sobreparametrizado. La idea clave es la existencia de dos zonas de actuación del modelo claramente diferenciadas, la zona infraparametrizada y la zona sobreparametrizada. Sin embargo, formalizar esta idea no es trivial, dado que los modelos profundos funcionan como «cajas negras», lo que hace que, a medida que aumenta su complejidad, resulte cada vez más difícil interpretar y analizar su funcionamiento interno.

Aunque cada vez hay más estudios abordando este hecho, muchos de ellos se centran en una perspectiva empírica del mismo, sin ofrecer una base teórica suficiente, mientras que otros sistematizan conceptos sin llegar a conclusiones prácticas. En este TFG, y con el objetivo de ofrecer una comprensión lo más completa posible, buscamos cerrar la brecha entre la teoría y la práctica, unificando explicaciones teóricas suficientemente rigurosas con ejemplos empíricos del mundo real.

## 1. Definición del problema

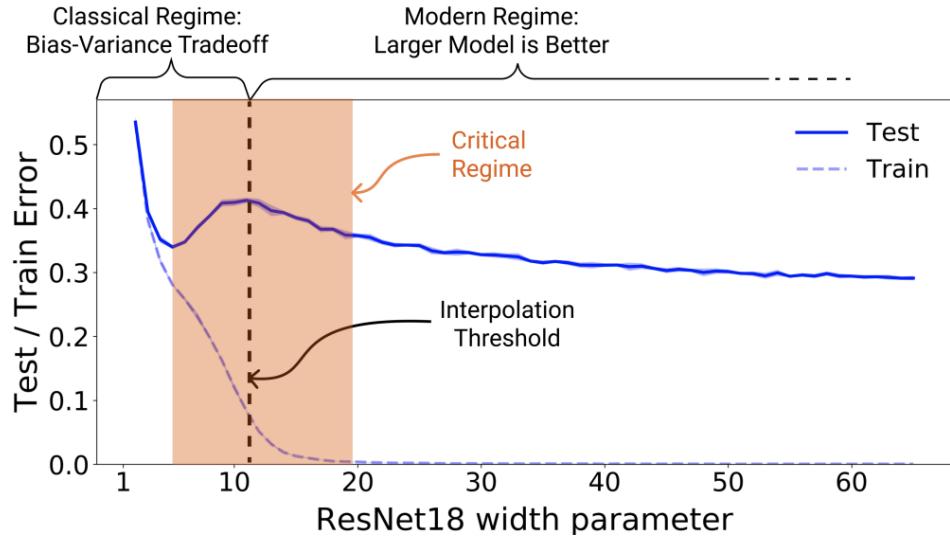


Figura 1.: Ejemplo de *Deep Double Descent* en ResNet18 [NKB<sup>+</sup>19]. La imagen muestra el error de entrenamiento (*train error*, curva discontinua) y de generalización (*test error*, curva continua) para la arquitectura ResNet18 con diferente capacidad (número de parámetros). En ella, observamos las tres regiones de actuación del modelo, así como el máximo del error de generalización correspondiente al umbral de interpolación y los dos descensos de dicho error.

El término **doble descenso profundo** (*Deep Double Descent* [BHMM19]) describe la forma que toma la curva del error de generalización (error fuera de la muestra, *out-of-sample error*) de un modelo de aprendizaje como función de la capacidad del mismo. De manera intuitiva, podemos distinguir tres zonas o regiones diferenciadas en dicha curva:

- **Región clásica (infraparametrizada):** En esta región, el modelo no puede capturar toda la complejidad subyacente de la distribución de los datos debido a su baja capacidad. Como resultado, el error de generalización disminuirá inicialmente, ligado al hecho de que el modelo aprende de los datos de entrenamiento. Sin embargo, llegado un momento, el error de generalización aumentará de manera progresiva (véase Figura 1, parte de la figura etiquetada como “Classical Regime: Bias-Variance Tradeoff”) dando lugar a la clásica curva en forma de “U”, relacionado con el hecho de que el modelo, en lugar de aprender patrones, memoriza los datos de entrenamiento.
- **Región moderna (sobreparametrizada):** En esta zona, representada en la Figura 1 bajo la etiqueta “Modern Regime: Larger Model is Better”, el modelo tiene una capacidad mayor de la necesaria para ajustar los datos de entrenamiento, es decir, dispone de suficientes herramientas (parámetros) para ajustar cada uno de los datos de entrenamiento. Contrariamente a lo que se esperaría según la sabiduría convencional, el error de generalización no necesariamente aumenta en este régimen y, bajo ciertas condiciones, dicho error puede reducirse nuevamente, produciendo un nuevo descenso de la curva del error de generalización que se conoce como doble descenso.

- **Región crítica:** Esta región, indicada en la Figura 1 como “Critical Regime”, marca la transición entre la región infraparametrizada y la región sobreparametrizada y engloba zonas de ambas regiones. Dentro de ella, se encuentra el llamado umbral de interpolación o *interpolation threshold*, que corresponde al punto donde el modelo tiene justo la capacidad suficiente para ajustar de manera prácticamente perfecta los datos de entrenamiento. En este punto crítico, el error de generalización alcanzará su máximo.

Este doble descenso puede llegar a suponer la obtención de modelos cuyas predicciones sean aún mejores. Sin embargo, se abre la puerta a la investigación del por qué ocurre este novedoso fenómeno, además de tener que replantearnos algunas respuestas, asumidas tradicionalmente como correctas, ante preguntas clásicas del aprendizaje profundo.

## 2. Motivación

En el ámbito del aprendizaje automático, existe una brecha entre el desarrollo empírico y la fundamentación teórica subyacente. Los modelos modernos, en particular las redes neuronales profundas, han demostrado resultados sorprendentes [Sej20, HT20] desde la generación de imágenes realistas mediante redes generativas [EEAMT22, RH21] hasta el procesamiento de lenguaje natural (*Natural Language Processing, NLP*) [KLW19, LLA22]. Sin embargo, estos resultados se logran sin una adecuada comprensión teórica [BD09, GK22].

Esta capacidad para obtener resultados que consideramos satisfactorios ha llevado a un enfoque predominantemente práctico, donde los avances se suelen producir de manera más rápida a través del ensayo y error y no tanto a través de modelos matemáticos bien fundamentados, lo que impide comprender, desde una perspectiva teórica, tanto la eficacia como las verdaderas limitaciones de estos modelos. El doble descenso plantea cuestiones sobre la sostenibilidad de este enfoque práctico y la necesidad de desarrollar un marco teórico que permita anticipar y guiar estos avances en lugar de simplemente reaccionar ante ellos.

Aunque existen elogiosos esfuerzos para comprender las bases teóricas del *deep learning* [ZBH<sup>+</sup>21, Mal16, Pri23, BB24, GK22, BB18, SBD<sup>+</sup>19], la desconexión entre avances teóricos y prácticos continúa siendo notable. Por ello, en los últimos años han ido apareciendo discrepancias, y se han observado fenómenos de carácter práctico, que han desafiado el conocimiento teórico tradicional. Es aquí donde se enmarca el concepto de *Deep Double Descent*.

De igual manera, este proyecto tiene una relevancia crucial, ya que podría transformar la forma en que se diseñan y optimizan los modelos profundos. Tradicionalmente, el enfoque clásico sugiere que, a medida que se aumenta la complejidad del modelo, este tiende a sobreajustarse a los datos, lo que limita su capacidad de generalización frente a nuevos datos no vistos y motivaba a optar por modelos más simples o incorporar estrategias de regularización. Sin embargo, con la aparición del *Deep Double Descent*, se ha demostrado que, más allá del sobreajuste, los modelos no solo proporcionan mejores predicciones, sino que incluso alcanzan niveles de generalización superiores a los iniciales. Este descubrimiento sugiere que, siguiendo este enfoque, podríamos prescindir de la teoría clásica y centrarnos en desarrollar modelos más complejos. No obstante, las limitaciones computacionales y energéticas siguen representando un desafío en el entrenamiento de grandes modelos de aprendizaje profundo [TGLM22, CRF<sup>+</sup>25, DMPHO23, SGM20]. Por ello, en la actualidad, aún se aplican técnicas para mitigar el sobreajuste y optimizar el uso de recursos.

En conclusión, el *Deep Double Descent* representa un cambio de paradigma, digno de estudio, en el aprendizaje automático. Comprender los principios subyacentes permitiría avanzar

en nuestra comprensión teórica del aprendizaje automático, reduciendo la brecha teórico-práctica, e incluso podría contribuir a guiar el diseño y optimización de modelos prácticos más efectivos.

### **3. Objetivos**

Los primeros indicios del *Deep Double Descent* se remontan a la década de 1990 y principios de los años 2000 [VCR89, Oppo1], aunque no se utilizaba específicamente esa terminología. Estos estudios mostraban la relación entre la complejidad del modelo y el error de generalización, indicando que, en algunos casos, aumentar la complejidad del modelo más allá de cierto punto no incrementaba necesariamente el error de generalización. No obstante, no es hasta 2019 cuando Belkin et al. [BHMM19] abordan la primera investigación formal sobre este tema y le asignan su particular nombre. A partir de ese momento, la comunidad científica comienza a mostrar un creciente interés hasta el día de hoy, lo que nos lleva a catalogarlo como un acontecimiento novedoso.

Por tanto, el objetivo principal de este TFG radica en tratar de **ofrecer una explicación detallada y estructurada del reciente concepto del Deep Double Descent**. Este estudio se centrará en proporcionar una visión actualizada y rigurosa de sus fundamentos teóricos, implicaciones prácticas y relevancia en el desarrollo de modelos modernos, asegurando que el contenido se mantenga en concordancia con los avances más recientes en la investigación. Para alcanzar este objetivo, se han definido dos líneas de trabajo profundamente interrelacionadas: una orientada al desarrollo **matemático** y otra enfocada a la parte **informática**. Ambas se desarrollan de manera conjunta y complementaria, de modo que los avances teóricos guían la implementación práctica, mientras que los resultados experimentales permiten validar y enriquecer la comprensión teórica. A su vez, cada línea de trabajo se descompone en una serie de objetivos parciales que, en conjunto, dirigen el desarrollo del proyecto.

#### **3.1. Objetivo matemático**

El objetivo fundamental para la parte matemática consiste en profundizar en la comprensión teórica del *Deep Double Descent* a través del estudio detallado de sus fundamentos matemáticos, explorando las relaciones con conceptos clásicos como el equilibrio sesgo-varianza y su posible conexión con la teoría de la aproximación no lineal. Con el fin de abordar de forma sistemática las distintas fases de este análisis, el presente objetivo se descompondrá en los siguientes objetivos parciales:

- Realizar un análisis exhaustivo y detallado del estado del arte, revisando las principales teorías, descubrimientos y avances matemáticos relacionados.
- Presentar de manera detallada las teorías y enfoques tradicionales que, a día de hoy, prevalecen en la literatura del aprendizaje automático.
- Investigar y analizar el *Deep Double Descent*, proporcionando una explicación detallada de sus fundamentos y explorando en profundidad los hallazgos más relevantes de la literatura científica.
- Adentrarnos en la teoría de la aproximación no lineal con el propósito de identificar y analizar posibles analogías, explorando cómo los enfoques no lineales pueden ofrecer una comprensión más profunda y enriquecedora del fenómeno.

### 3.2. Objetivo informático

El objetivo esencial para la parte informática consiste en llevar a cabo la constatación experimental del *Deep Double Descent* mediante la implementación y análisis de diversas arquitecturas que permitan validar y estudiar empíricamente este comportamiento. Esta parte experimental busca no solo ilustrar su aparición en distintos escenarios y arquitecturas, sino también corroborar y complementar los resultados obtenidos en la parte matemática, estableciendo así una conexión sólida entre la teoría y la práctica. Con el fin de abordar de forma sistemática las distintas fases de este análisis, el presente objetivo se descompondrá en los siguientes objetivos parciales:

- Llevar a cabo un estudio profundo y minucioso de los casos prácticos en los que se ha manifestado, revisando los principales comportamientos y patrones.
- Presentar resultados experimentales que validen los desarrollos teóricos realizados en la parte matemática, demostrando la coherencia con las predicciones teóricas.
- Desarrollar un análisis experimental que respalte la aparición y las características del *Deep Double Descent*, proporcionando evidencias prácticas que contribuyan a una comprensión más profunda de su comportamiento en diferentes modelos y escenarios.

## 4. Planificación del proyecto

De cara a planificar el proyecto, es fundamental considerar que el TFG en el doble grado en Ingeniería Informática y Matemáticas consta de 18 créditos ECTS. Teniendo en cuenta que cada ECTS es equivalente a 25 horas de trabajo, se estima que se necesitarán, de manera teórica, un total de 450 horas para la realización del mismo. Debido al estrecho vínculo entre la informática y las matemáticas en este proyecto, el análisis integra ambos aspectos, considerando el total de horas dedicadas.

Dada la distribución temporal del segundo cuatrimestre, con aproximadamente 20 semanas disponibles, se estima que la realización del proyecto requerirá 30 horas semanales, equivalentes a 5 horas diarias durante 6 días a la semana. Esto se traduce en 120 horas al mes que, durante un lapso de aproximadamente 4 meses, suman un total de 480 horas. Por tanto, se reservan 2 semanas como margen para posibles imprevistos que puedan surgir durante el desarrollo del proyecto.

Inicialmente, el TFG se estructuró siguiendo una metodología basada en el ciclo de vida en cascada [Pre94]. Sin embargo, este enfoque impide retroceder entre fases, lo que puede dificultar la adaptación a problemas o cambios identificados tras completar una etapa. Aunque el proyecto cuenta con requisitos y objetivos bien definidos, es común que surjan ajustes necesarios. Por ello, se emplea un modelo en cascada con retroalimentación, que permite revisar y modificar fases anteriores cuando sea necesario (véase Figura 2).

El proyecto se organiza en las siguientes fases del ciclo de vida:

- Análisis de requisitos: Consiste en las reuniones iniciales con los clientes, en este caso los directores del TFG. Se realiza un estudio de la bibliografía existente, se establecen los objetivos del trabajo y se traza un camino claro de los resultados que se quieren alcanzar.

## Introducción

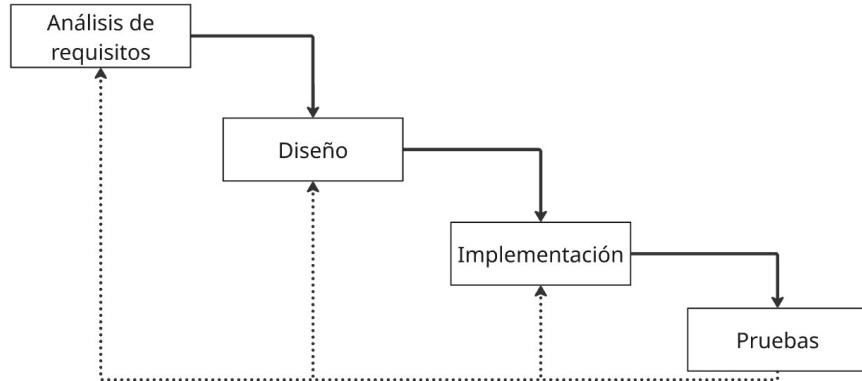


Figura 2.: Modelo en cascada con realimentación. Las flechas continuas representan el flujo principal del proceso en cascada, mientras que las flechas punteadas indican las fases de realimentación.

- **Diseño:** Consiste en la investigación y selección de técnicas aplicables a la resolución del problema, incluyendo los conjuntos de datos y modelos a utilizar. En este apartado también se incluyen diversas pruebas preliminares y el diseño del software utilizado en la experimentación.
- **Implementación:** Consiste en la adaptación del código de los modelos investigados en la fase anterior, así como la implementación de nuevas funcionalidades.
- **Pruebas:** Consiste en la realización de diversos experimentos utilizando los conjuntos de datos y modelos previamente definidos. En esta fase se contempló (y efectivamente se llevó a cabo) la posibilidad de regresar a la fase de diseño, ya que, aunque algunos experimentos se definen inicialmente, la experimentación puede revelar nuevas posibilidades que deben ser evaluadas.

Además, de manera paralela a las etapas descritas anteriormente, se lleva a cabo un proceso continuo de redacción y documentación de la memoria del proyecto, en el que se formalizan todas las notas e investigaciones realizadas para cada capítulo, así como el detalle de cada decisión tomada durante el desarrollo del proyecto.

Tareas	Semanas - Horas	Enero		Febrero			Marzo				Abril			Mayo			Junio				
		20	27	03	10	17	24	03	10	17	24	31	07	14	21	28	05	12	19	26	02
Analisis de requisitos	5 – 150																				
Diseño	4 – 120																				
Implementación	2 – 60																				
Pruebas	7 – 210																				
Documentación	–																				

Tabla 1.: Planificación temporal inicial del proyecto. La documentación se desarrolla de forma transversal a lo largo de la duración del proyecto, en paralelo al resto de fases.

La planificación inicial del proyecto se detalla en la Tabla 1. No obstante, debido al carácter novedoso del trabajo y a los continuos avances por parte de la comunidad científica a lo largo de este año, la planificación inicial experimentó algunas modificaciones. Estos ajustes estuvieron motivados por nuevos descubrimientos que permitieron comprender el fenómeno con mayor claridad, lo que obligó a retroceder a la fase inicial del proyecto y prolongó su

#### 4. Planificación del proyecto

Tareas	Semanas - Horas	Enero		Febrero			Marzo				Abril			Mayo			Junio				
		20	27	03	10	17	24	03	10	17	24	31	07	14	21	28	05	12	19	26	02
Analisis de requisitos	8 – 240																				
Diseño	4 – 120																				
Implementación	2 – 60																				
Pruebas	7 – 210																				
Documentación	–																				

Tabla 2.: Planificación temporal final del proyecto. La documentación se desarrolla de forma transversal a lo largo de la duración del proyecto, en paralelo al resto de fases.

duración. De este modo, la planificación final del proyecto se presenta en la Tabla 2, mientras que en la Tabla 3 se presenta un resumen de la duración total.

Fecha de inicio	20/01/2025
Fecha de fin	02/05/2025
Duración	133 días, 95 laborables

Tabla 3.: Resumen de las fechas de inicio y fin del proyecto, junto con la duración total.

Para la estimación del coste, partimos de la base de que el coste por hora de un investigador senior o responsable de I+D en una empresa tecnológica española es de 15€/hora, según el portal de transparencia empresarial Glassdoor<sup>1</sup>, y de una duración final del proyecto de 630 horas. A esta cifra se deben sumar los gastos derivados de los distintos materiales utilizados, tales como el coste del portátil empleado en el desarrollo del TFG y el uso de un servidor GPU de altas prestaciones, junto a otros gastos misceláneos entre los que se incluye el consumo eléctrico. El desglose detallado de estos costes se puede consultar en la Tabla 4.

Respecto al servidor GPU, y basándonos en sus especificaciones, se estima su valoración en 12000€. Se asume una amortización proyectada a lo largo de dos años, lo que equivale a un coste diario de 16.44€. Por ende, la contribución total de este servidor al coste del proyecto sería de 2186.52€.

Item	Costo
Salario	9 450.00€
Portátil de Gama Media	1 000.00€
Servidor GPU	2 186.52€
Otros	300.00€
<b>Total</b>	<b>12 936.52€</b>

Tabla 4.: Estimación del coste del proyecto.

<sup>1</sup>Se asume un salario anual de 30000€ según estimaciones publicadas en: [https://www.glassdoor.es/Sueldos/ingeniero-de-investigacion-y-desarrollo-sueldo-SRCH\\_K00\\_39.htm](https://www.glassdoor.es/Sueldos/ingeniero-de-investigacion-y-desarrollo-sueldo-SRCH_K00_39.htm)



# **Parte I.**

## **Fundamentos Teóricos**



# 1. Probabilidad

En este capítulo se presentarán algunas definiciones y resultados de la teoría de la probabilidad y la estadística, con el propósito de introducir conceptos clave que faciliten la comprensión del *Deep Double Descent* y que utilizaremos a lo largo del desarrollo de gran parte del trabajo. Las fuentes principales utilizadas a lo largo de este capítulo son extractos de [Dem14, Knio9].

## 1.1. Espacios de probabilidad y $\sigma$ -álgebras

Para establecer la base teórica, consideraremos un conjunto arbitrario  $\Omega$ , al que nos referiremos como **espacio muestral** y que representa el conjunto de todos los posibles resultados al realizar un experimento. Asimismo, llamaremos **suceso** a cualquier subconjunto de  $\Omega$ .

**Definición 1.1** ( $\sigma$ -álgebra). Un conjunto  $\mathcal{A}$  de subconjuntos de  $\Omega$  ( $\mathcal{A} \subseteq \mathcal{P}(\Omega)$ ) se dirá que es una  $\sigma$ -álgebra si verifica las siguientes propiedades:

1.  $\Omega \in \mathcal{A}$ .
2. Si  $A \in \mathcal{A}$ , entonces  $A^c = \Omega \setminus A \in \mathcal{A}$  ( $A$  es cerrado bajo complementarios).
3. Si  $A_n \in \mathcal{A}$ , entonces  $\bigcup_{n \in \mathbb{N}} A_n \in \mathcal{A}$  ( $A$  es cerrado bajo uniones finitas).

Es fácil comprobar que  $\mathcal{A}$  es cerrado bajo intersecciones finitas y que, además, la intersección de  $\sigma$ -álgebras es una  $\sigma$ -álgebra.

**Definición 1.2** ( $\sigma$ -álgebra de Borel). Para cada conjunto  $\mathcal{C}$  de subconjuntos de  $\Omega$ , se define  $\sigma(\mathcal{C})$  como la menor  $\sigma$ -álgebra  $\mathcal{A}$  que contiene a  $\mathcal{C}$ . La  $\sigma$ -álgebra  $\mathcal{A}$  es la intersección de todas las  $\sigma$ -álgebras que contienen a  $\mathcal{C}$  y, por tanto, es una  $\sigma$ -álgebra.

Si  $(E, \mathcal{O})$  es un espacio topológico, donde  $\mathcal{O}$  es el conjunto formado por los conjuntos abiertos en  $E$ , entonces  $\sigma(\mathcal{O})$  es llamada la  **$\sigma$ -álgebra de Borel** del espacio topológico.

Llamaremos **espacio de medida** al conjunto  $(\Omega, \mathcal{A})$  donde  $\mathcal{A}$  es una  $\sigma$ -álgebra en  $\Omega$ .

**Definición 1.3** (Medida de probabilidad). Dado un espacio de medida  $(\Omega, \mathcal{A})$ , una función  $P : \mathcal{A} \rightarrow \mathbb{R}$  se llamará *medida de probabilidad* si cumple las siguientes tres propiedades (conocidas como **axiomas de Kolmogorov** [Kol56]):

1.  $P[A] \geq 0$  para todo  $A \in \mathcal{A}$ .
2.  $P[\Omega] = 1$ .
3.  $P$  es  $\sigma$ -aditiva, es decir, si  $A_n \in \mathcal{A}$  con  $n \in \mathbb{N}$  son conjuntos disjuntos dos a dos, entonces

$$P \left[ \bigcup_{n \in \mathbb{N}} A_n \right] = \sum_{n \in \mathbb{N}} P[A_n].$$

## 1. Probabilidad

La primera condición nos asegura la no negatividad de la probabilidad, es decir, la probabilidad nunca será inferior a 0. A su vez, la segunda condición nos establece que la probabilidad del espacio muestral completo ( $\Omega$ ) debe ser igual a 1, es decir, refleja que uno de los eventos posibles siempre ocurrirá, conocido como **suceso seguro**.

*Observación 1.4.* Como consecuencia de las propiedades 2 y 3 de la definición anterior, se sigue de manera inmediata que  $P(\emptyset) = 0$ . Además, al conjunto  $\emptyset$  se le suele denominar como *evento imposible*.

**Corolario 1.5.** *Algunas propiedades básicas de la medida de probabilidad ( $P$ ) que se siguen de la propia definición son las siguientes:*

1. Si  $A, B \in \mathcal{A}$  y  $A \subset B$ , entonces  $P[A] \leq P[B]$ .
2.  $P[A^c] = 1 - P[A]$ , para todo  $A \in \mathcal{A}$ .
3.  $0 \leq P[A] \leq 1$ , para todo  $A \in \mathcal{A}$ .

*Observación 1.6.* Existen distintas formas de construir los axiomas para un espacio de probabilidad. Por ejemplo, se podrían sustituir las primeras dos propiedades de la definición de medida de probabilidad por las últimas dos propiedades enunciadas en el corolario anterior.

**Definición 1.7** (Espacio de probabilidad). Dado un espacio de medida  $(\Omega, \mathcal{A})$ , llamaremos *espacio de probabilidad* a la tripleta  $(\Omega, \mathcal{A}, P)$ , donde  $P$  es una medida de probabilidad.

## 1.2. Variables aleatorias y esperanza

Las variables aleatorias son funciones numéricas que asignan un valor numérico a cada posible resultado de un experimento aleatorio  $w \in \Omega$ . De manera intuitiva, una variable aleatoria puede verse como una cantidad numérica cuyo valor no es fijo y que puede tomar distintos valores, por lo que es necesario definir una distribución de probabilidad que asocie probabilidades a los distintos valores que pueda tomar la variable aleatoria.

**Definición 1.8** (Función medible). Una función  $X : (\Omega_1, \mathcal{A}) \rightarrow (\Omega_2, \mathcal{B})$  se dice *medible* si

$$X^{-1}(B) \in \mathcal{A}, \quad \forall B \in \mathcal{B},$$

donde el conjunto  $X^{-1}(B)$  se encuentra formado por todos los puntos  $x \in \Omega$  para los cuales  $X(x) \in B$ .

**Definición 1.9** (Variable aleatoria). Dado un espacio de probabilidad  $(\Omega_1, \mathcal{A}, P)$  y un espacio medible  $(\Omega_2, \mathcal{B})$ , decimos que  $X : (\Omega_1, \mathcal{A}, P) \rightarrow (\Omega_2, \mathcal{B})$  es una *variable aleatoria* si  $X$  es una función medible.

Además, si el espacio medible de llegada es  $n$ -dimensional, entonces la variable aleatoria  $X$  es llamada **vector aleatorio** y lo denotaremos por  $X = (X_1, X_2, \dots, X_n)$ , donde cada componente  $X_i$ , con  $i \in \{1, 2, \dots, n\}$ , es una variable aleatoria.

*Observación 1.10.* En la mayoría de usos prácticos se tiene que el espacio medible de llegada más común es  $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$ , donde  $\mathcal{B}(\mathbb{R})$  denota la  $\sigma$ -álgebra de Borel en  $\mathbb{R}$ .

**Definición 1.11.** Diremos que una variable aleatoria es *discreta* si esta toma un número finito o numerable de valores en el espacio de llegada. Por otra parte, si la variable aleatoria toma un número infinito o no numerable de valores, diremos que es una variable aleatoria *continua*.

**Ejemplo 1.12.** Como ejemplo sencillo podemos considerar los posibles resultados obtenidos al lanzar un dado de seis caras, es decir,  $w \in \{1, 2, 3, 4, 5, 6\}$ , y podemos definir la variable aleatoria discreta que asigna a cada resultado el valor de la cara superior obtenida al realizar el lanzamiento. En este caso, la variable aleatoria se define como:

$$X(w) = w, \quad \text{para } w \in \{1, 2, 3, 4, 5, 6\}.$$

**Definición 1.13** (Función de distribución). La *función de distribución (acumulada)* de una variable aleatoria  $X$  es la función  $F_X : \mathbb{R} \rightarrow [0, 1]$  definida por:

$$F_X(\alpha) = P[\{w : X(w) \leq \alpha\}], \quad \forall \alpha \in \mathbb{R}.$$

**Proposición 1.14.** La función de distribución  $F$  de una variable aleatoria  $X$  cumple las siguientes propiedades:

1.  $F$  es monótona no decreciente.
2.  $\lim_{x \rightarrow \infty} F(x) = 1$  y  $\lim_{x \rightarrow -\infty} F(x) = 0$ .
3.  $F$  es continua por la derecha, es decir,  $\lim_{y \rightarrow x^+} F(y) = F(x)$ .

**Definición 1.15** (Función de probabilidad). Sea  $X$  una variable aleatoria discreta, llamaremos *función (masa) de probabilidad* a la función que asigna la probabilidad de que la variable aleatoria tome un valor en particular, es decir:

$$p_X(x) = P[X = x], \quad \text{con } x \in \{x_1, \dots, x_n\}.$$

Las probabilidades asociadas con todos los posibles resultados del experimento deben ser no negativas y sumar uno, es decir  $\sum_x p_X(x) = 1$  y, además,  $p_X(x) \geq 0$ .

Notemos que el concepto de función de probabilidad solo tiene sentido al hablar de variables aleatorias discretas. Para variables aleatorias continuas, el concepto análogo es el de función de densidad, donde deberemos integrar para obtener la probabilidad, pues la probabilidad asociada a un único punto en un intervalo es cero.

**Definición 1.16** (Función de densidad). Se dice que una función  $f_X$ , integrable de Lebesgue y no negativa en casi todas partes, es la *función de densidad* de una variable aleatoria continua  $X$  si su función de distribución puede ser expresada como

$$F_X(\alpha) = \int_{-\infty}^{\alpha} f_X(x) dx, \quad \forall \alpha \in \mathbb{R}.$$

Notemos que la función de densidad, de manera análoga a la función de probabilidad, cumple  $f_X(x) \geq 0$  y  $\int_{-\infty}^{\infty} f_X(x) dx = 1$ .

**Definición 1.17 (Esperanza de una variable aleatoria).** Sea  $X$  una variable aleatoria en el espacio de probabilidad  $(\Omega, \mathcal{A}, P)$ . Definimos el **valor esperado o esperanza** de  $X$ , denotada por  $E[X]$ , como la integral de Lebesgue siguiente:

$$\mathbb{E}[X] = \int_{\Omega} X(w) dP[w].$$

Para vectores aleatorios, su esperanza viene definida componente a componente:

$$\mathbb{E}[(X_1, \dots, X_n)] = (\mathbb{E}[X_1], \dots, \mathbb{E}[X_n]).$$

## 1. Probabilidad

*Observación 1.18.* Si  $X$  es una variable aleatoria discreta con función de probabilidad  $P[X = x_i]$ , con  $i \in \{1, 2, \dots, n\}$ , su esperanza viene definida por:

$$\mathbb{E}[X] = \sum_{i=1}^n x_i P[X = x_i],$$

donde  $x_i$  denota cada posible resultado del experimento.

*Observación 1.19.* Si  $X$  es una variable aleatoria continua con función de densidad  $f_X(x)$ , su esperanza se define como:

$$\mathbb{E}[X] = \int_{\mathbb{R}} x f_X(x) dx.$$

### 1.2.1. Probabilidad condicional

En esta sección introduciremos la noción clásica de *probabilidad condicional* de sucesos, ligada al supuesto de conocer la probabilidad de un cierto suceso bajo la condición de que ocurra otro suceso. De igual manera, se introducirán resultados necesarios para el desarrollo del trabajo, tales como el teorema de Bayes.

**Definición 1.20** (Probabilidad condicional). Dados dos sucesos  $A, B \in \mathcal{A}$  con  $P[B] > 0$ , definimos la *probabilidad condicional* de  $A$  con respecto a  $B$  de la siguiente forma:

$$P[A|B] = \frac{P[A \cap B]}{P[B]}.$$

**Definición 1.21.** Un conjunto finito  $\{A_1, \dots, A_n\} \subset \mathcal{A}$  se denominará *partición finita* de  $\Omega$  si cumple:

1.  $\bigcup_{i=1}^n A_i = \Omega$ .
2.  $A_i \cap A_j = \emptyset$ , para todo  $i \neq j$ .

Una partición finita cubre todo el espacio con un número finito de conjuntos (sucesos) disjuntos dos a dos.

**Teorema 1.22** (Probabilidad total). *Sean  $\{A_1, \dots, A_n\}$  una partición finita de  $\mathcal{A}$  y  $B \in \mathcal{A}$  un suceso cualquiera del que se conocen las probabilidades condicionales  $P[B|A_i] \forall i \in \{1, \dots, n\}$ . Entonces, la probabilidad del suceso  $B$  viene dada por la siguiente expresión:*

$$P[B] = \sum_{i=1}^n P[B|A_i]P[A_i].$$

*Demostración.* Partimos de una partición finita  $\{A_1, \dots, A_n\}$  de  $\mathcal{A}$  y de un suceso  $B \in \mathcal{A}$ . Usando la primera propiedad de la Definición 1.21, podemos expresar el suceso  $B$  de la siguiente forma:

$$B = (B \cap A_1) \cup (B \cap A_2) \cup \dots \cup (B \cap A_n).$$

Utilizando la segunda propiedad de la Definición 1.21, sabemos que  $A_i \cap A_j = \emptyset, i \neq j$ . Por tanto, obtenemos que los conjuntos  $(B \cap A_i), i \in \{1, \dots, n\}$  son, también, disjuntos dos a dos.

Por consiguiente, podemos expresar la probabilidad del suceso  $B$  como sigue:

$$P[B] = P[B \cap A_1] + P[B \cap A_2] + \cdots + P[B \cap A_n].$$

A continuación, usando la Definición 1.20, obtenemos que

$$P[A_i \cap B] = P[A_i|B]P[B], \quad \forall i \in \{1, \dots, n\}.$$

Finalmente, obtenemos la expresión buscada:

$$\begin{aligned} P[B] &= P[B \cap A_1] + P[B \cap A_2] + \cdots + P[B \cap A_n] \\ &= P[B|A_1]P[A_1] + P[B|A_2]P[A_2] + \cdots + P[B|A_n]P[A_n] \\ &= \sum_{i=1}^n P[B|A_i]P[A_i]. \end{aligned}$$

□

Llegados a este punto estamos en las condiciones necesarias de introducir un teorema fundamental que nos permite calcular probabilidades condicionales. Este resultado vincula la probabilidad de un suceso  $A$  dado otro suceso  $B$  ( $P[A|B]$ ) con la probabilidad del suceso  $B$  dado el suceso  $A$  ( $P[B|A]$ ).

**Teorema 1.23** (Regla de Bayes). *Dada una partición finita  $\{A_1, \dots, A_n\}$  de  $\mathcal{A}$  y un suceso  $B \in \mathcal{A}$  con  $P[B] > 0$ , se verifica:*

$$P[A_i|B] = \frac{P[B|A_i]P[A_i]}{\sum_{i=1}^n P[B|A_i]P[A_i]}.$$

*Demostración.* En primer lugar, de la Definición 1.20, sabemos que  $P[A_i|B] = \frac{P[A_i \cap B]}{P[B]}$ . Además, del Teorema 1.22, conocemos que  $P[B] = \sum_{i=1}^n P[B|A_i]P[A_i]$ .

Por otra parte, podemos aplicar la Definición 1.20 de la siguiente manera:

$$P[B|A_i] = \frac{P[B \cap A_i]}{P[A_i]}.$$

Ahora, despejando, obtenemos  $P[A_i \cap B] = P[B \cap A_i] = P[B|A_i]P[A_i]$ . Finalmente, combinando ambos resultados alcanzamos la conclusión buscada:

$$P[A_i|B] = \frac{P[A_i \cap B]}{P[B]} = \frac{P[B|A_i]P[A_i]}{\sum_{i=1}^n P[B|A_i]P[A_i]}.$$

□

## 1.2.2. Independencia de variables aleatorias

En esta sección nos centraremos en explicar el concepto de independencia en el contexto de variables aleatorias. De manera intuitiva, el concepto de independencia, como su nombre indica, va ligado al hecho de que el conocimiento que poseamos de una de las variables no proporciona información adicional sobre el conocimiento de la otra. Para ello, también expandiremos el concepto de función de distribución para el caso de más de una variable aleatoria (vector aleatorio).

## 1. Probabilidad

**Definición 1.24** (Función de distribución conjunta). Sean  $X_1, \dots, X_n$  variables aleatorias definidas sobre el mismo espacio de probabilidad  $(\Omega, \mathcal{A}, P)$ , la *función de distribución conjunta* es la función  $F_{X_1, \dots, X_n} : \mathbb{R} \rightarrow [0, 1]$  definida por

$$F_{X_1, \dots, X_n}(x_1, \dots, x_n) = P[X_1 \leq x_1, \dots, X_n \leq x_n], \quad x_1, \dots, x_n \in \mathbb{R}.$$

Si interpretamos las  $n$  variables aleatorias como un vector aleatorio  $X = (X_1, \dots, X_n)$ , podemos simplificar la notación de la siguiente forma:

$$F_{X_1, \dots, X_n}(x_1, \dots, x_n) = F_X(x_1, \dots, x_n).$$

De la misma forma, podemos extender las definiciones de *función de probabilidad* y *función de densidad* para el caso en el que dispongamos de más de una variable aleatoria.

En primer lugar, comenzaremos definiendo y demostrando el resultado de la **regla de la cadena** para probabilidades, que nos será de gran utilidad para trabajar con las siguientes definiciones.

**Teorema 1.25** (Regla de la cadena). *Sea  $(\Omega, \mathcal{A}, P)$  un espacio de probabilidad y  $\{A_1, \dots, A_n\} \in \mathcal{A}$  una serie de sucesos. Entonces, se verifica*

$$P[A_1 \cap A_2 \cap \dots \cap A_n] = P[A_1]P[A_2|A_1] \cdots \prod_{i=2}^n P[A_i|A_1 \cap \dots \cap A_{i-1}].$$

*Demostración.* La demostración se realiza de manera sencilla por recursión, teniendo en cuenta que en el primer caso se hace uso de la Definición 1.20:

$$P[A_1 \cap A_2] = P[A_1]P[A_2|A_1].$$

□

**Definición 1.26** (Función de probabilidad conjunta). Sean  $X_1, \dots, X_n$  variables aleatorias discretas. La *función (masa) de probabilidad conjunta* de dichas variables viene dada por

$$p_{X_1, \dots, X_n}(x_1, \dots, x_n) = P[X_1 = x_1, \dots, X_n = x_n], \quad (x_1, \dots, x_n) \in \mathbb{R}^n.$$

Equivalentemente, la función de probabilidad conjunta puede ser expresada de la siguiente forma:

$$\begin{aligned} p_{X_1, \dots, X_n}(x_1, \dots, x_n) &= P(X_1 = x_1) \cdot P(X_2 = x_2 | X_1 = x_1) \cdot \\ &\quad \cdot P(X_3 = x_3 | X_1 = x_1, X_2 = x_2) \cdots \\ &\quad \cdot P(X_n = x_n | X_1 = x_1, X_2 = x_2, \dots, X_{n-1} = x_{n-1}), \end{aligned}$$

donde se utiliza la definición de probabilidad condicionada y la regla de la cadena comentada en el teorema anterior.

Además, dado que estamos trabajando con probabilidades, se verifica que la suma total debe ser igual a uno, es decir

$$\sum_i \sum_j \cdots \sum_k P[X_1 = x_{1i}, X_2 = x_{2j}, \dots, X_n = x_{nk}] = 1,$$

donde los distintos índices  $(i, j, \dots, k)$  recorren todos los posibles resultados de cada variable aleatoria.

**Definición 1.27** (Función de densidad conjunta). Sean  $X_1, \dots, X_n$  variables aleatorias continuas. Se dice que una función  $f_{X_1, \dots, X_n}$ , integrable de Lebesgue y no negativa en casi todas partes, es la *función de densidad conjunta* de las variables aleatorias continuas  $X_1, \dots, X_n$  si la función de distribución conjunta puede ser expresada como

$$F_{X_1, \dots, X_n}(x_1, \dots, x_n) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_n} f_{X_1, \dots, X_n}(x_1, \dots, x_n) dx_1 \cdots dx_n, \\ \forall (x_1, \dots, x_n) \in \mathbb{R}^n.$$

Una manera análoga de expresar la función de densidad conjunta es la siguiente:

$$f_{X_1, \dots, X_n}(x_1, \dots, x_n) = \frac{\partial^n F_{X_1, \dots, X_n}(x_1, \dots, x_n)}{\partial x_1 \cdots \partial x_n}.$$

De manera análoga a la definición anterior y dado que estamos trabajando con distribuciones de probabilidad, se verifica que

$$\int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_n} f_{X_1, \dots, X_n}(x_1, \dots, x_n) dx_1 \cdots dx_n = 1.$$

Una vez conocemos las funciones conjuntas de una serie de variables aleatorias, es posible determinar la probabilidad de un suceso sin considerar la influencia de otras variables por medio de las funciones marginales.

**Definición 1.28** (Función de distribución marginal). Sean  $X_1, \dots, X_n$  variables aleatorias. Se define la *función de probabilidad marginal* de la variable aleatoria  $X_i$ , para  $i \in \{1, \dots, n\}$ , de la siguiente manera:

$$\forall i = \{1, \dots, n\}, \quad F_{X_i}(x_i) = F_{X_1, \dots, X_n}(+\infty, \dots, x_i, \dots, +\infty), \quad \forall x_i \in \mathbb{R}.$$

**Definición 1.29** (Función de probabilidad marginal). Sean  $X_1, \dots, X_n$  variables aleatorias discretas. Se define la *función (masa) de probabilidad marginal* de la variable aleatoria  $X_i$ , para  $i \in \{1, \dots, n\}$ , de la siguiente forma:

$$p_{X_i}(x_i) = \sum_{x_1} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_n} p_{X_1, \dots, X_n}(x_1, \dots, x_n), \quad (x_1, \dots, x_n) \in \mathbb{R}.$$

**Definición 1.30** (Función de densidad marginal). Sean  $X_1, \dots, X_n$  variables aleatorias continuas. Se define la *función de densidad marginal* de la variable aleatoria  $X_i$ , para  $i \in \{1, \dots, n\}$ , como sigue:

$$f_{X_i}(x_i) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_n} f_{X_1, \dots, X_n}(x_1, \dots, x_n) dx_1 \cdots dx_{i-1} dx_{i+1} \cdots dx_n, \quad \forall x_i \in \mathbb{R}.$$

**Definición 1.31.** Sean  $X_1, \dots, X_n$  variables aleatorias definidas sobre el mismo espacio de probabilidad  $(\Omega, \mathcal{A}, P)$ , con funciones de distribución  $F_{X_1}, \dots, F_{X_n}$  respectivamente. Decimos que las variables aleatorias son **idénticamente distribuidas (i.d.)** si

$$F_{X_1}(x) = F_{X_j}(x), \quad \forall j \in \{1, \dots, n\} \text{ y } \forall x \in \mathbb{R}.$$

## 1. Probabilidad

**Definición 1.32.** Sean  $X_1, \dots, X_n$  variables aleatorias definidas sobre el mismo espacio de probabilidad  $(\Omega, \mathcal{A}, P)$ , con funciones de distribución  $F_{X_1}, \dots, F_{X_n}$  respectivamente y función de distribución conjunta  $F_X$ . Decimos que las variables aleatorias son **independientes** si

$$F_X(x_1, \dots, x_n) = F_{X_1}(x_1) \cdots F_{X_n}(x_n), \quad \forall x_1, \dots, x_n \in \mathbb{R}.$$

*Observación 1.33.* De la definición anterior se deduce que dos variables aleatorias son independientes cuando:

- En el caso discreto, su función (masa) de probabilidad conjunta es igual al producto de las funciones (masa) de probabilidad marginales de cada variable aleatoria. Esto es, si  $X_1, X_2$  son dos variables aleatorias discretas, entonces  $p_{X_1, X_2}(x_1, x_2) = p_{X_1}(x_1)p_{X_2}(x_2)$ ,  $x_1, x_2 \in \mathbb{R}$ .
- En el caso continuo, su función de densidad conjunta es igual al producto de las funciones de densidad marginales de cada variable aleatoria. Esto es, si  $X_1, X_2$  son dos variables aleatorias continuas, entonces  $f_{X_1, X_2}(x_1, x_2) = f_{X_1}(x_1)f_{X_2}(x_2)$ ,  $x_1, x_2 \in \mathbb{R}$ .

**Definición 1.34.** Las variables aleatorias que cumplen, de manera simultánea, la Definición 1.31 y la Definición 1.32, las llamaremos *variables aleatorias independientes e idénticamente distribuidas* y se denotarán como *variables aleatorias (i.d.d.)*.

### 1.2.3. Propiedades de la esperanza y varianza

A continuación, se detallarán algunas de las propiedades fundamentales acerca de la esperanza de una variable aleatoria, que serán de gran utilidad de cara a realizar simplificaciones cuando trabajemos con variables aleatorias. Asimismo, se introduce el concepto de varianza, que está estrechamente relacionado con la esperanza y que usaremos de forma recurrente en el devenir del trabajo.

**Proposición 1.35.** La esperanza matemática de una constante ( $k \in \mathbb{R}$ ) para una variable aleatoria  $X$  es la propia constante.

*Demostración.*

$$\mathbb{E}[k] = \int_{-\infty}^{+\infty} k f_X(x) dx = k \cdot \int_{-\infty}^{+\infty} f_X(x) dx = k,$$

pues  $f_X(x)$  es función de densidad de la variable aleatoria  $X$  y, por tanto, el valor de su integral es 1.  $\square$

**Proposición 1.36** (Linealidad de la esperanza). Sean  $X, Y$  dos variables aleatorias y  $\alpha, \beta \in \mathbb{R}$ . Se tiene que  $\mathbb{E}[X]$  es un operador lineal, es decir:

$$\mathbb{E}[\alpha X + \beta Y] = \alpha \mathbb{E}[X] + \beta \mathbb{E}[Y].$$

*Demostración.* La demostración es consecuencia trivial de la linealidad de la integral de Lebesgue.  $\square$

**Teorema 1.37.** Sean  $X_1, \dots, X_n$  variables aleatorias independientes definidas sobre el mismo espacio de probabilidad  $(\Omega, \mathcal{A}, P)$ , tales que existe la esperanza de cada una de ellas, es decir,  $\exists \mathbb{E}[X_i] \forall i \in \{1, \dots, n\}$ . Entonces, existe  $\mathbb{E}[X_1 \cdots X_n]$  y, además, se verifica

$$\mathbb{E}[X_1 \cdots X_n] = \mathbb{E}[X_1] \cdots \mathbb{E}[X_n] = \prod_{i=1}^n \mathbb{E}[X_i].$$

*Demostración.* La demostración se sigue de manera sencilla utilizando el concepto de independencia. Para ello y por simplificar, consideramos el caso en el que tenemos dos variables aleatorias continuas, pues el resultado para el caso discreto y teniendo  $n$  variables aleatorias es análogo.

Por tanto, sean  $X_1, X_2$  dos variables aleatorias continuas con función de densidad asociada  $f_{X_1}$  y  $f_{X_2}$  respectivamente. La esperanza de la multiplicación de dichas variables viene dada por

$$\mathbb{E}[X_1 X_2] = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x_1 x_2 f_{X_1, X_2}(x_1, x_2) dx_1 dx_2,$$

donde  $f_{X_1, X_2}(x_1, x_2)$  denota la función de densidad conjunta de las variables aleatorias. Dado que las variables aleatorias son independientes, la función de densidad conjunta se puede expresar como el producto de las funciones marginales de cada variable aleatoria, es decir

$$f_{X_1, X_2}(x_1, x_2) = f_{X_1}(x_1) f_{X_2}(x_2).$$

Finalmente, sustituyendo este resultado en la expresión anterior obtenemos el resultado buscado:

$$\begin{aligned} \mathbb{E}[X_1 X_2] &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x_1 x_2 f_{X_1, X_2}(x_1, x_2) dx_1 dx_2 \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x_1 x_2 f_{X_1}(x_1) f_{X_2}(x_2) dx_1 dx_2 \\ &= \left( \int_{-\infty}^{+\infty} x_1 f_{X_1}(x_1) dx_1 \right) \left( \int_{-\infty}^{+\infty} x_2 f_{X_2}(x_2) dx_2 \right) \\ &= \mathbb{E}[X_1] \mathbb{E}[X_2]. \end{aligned}$$

□

**Proposición 1.38.** Sean  $X : (\Omega, \mathcal{A}, P) \rightarrow (\mathbb{R}, \mathcal{B}(\mathbb{R}))$  una variable aleatoria y  $g : (\mathbb{R}, \mathcal{B}(\mathbb{R})) \rightarrow (\mathbb{R}, \mathcal{B}(\mathbb{R}))$  una función (medible) integrable de Lebesgue, entonces  $g(X)$  es una variable aleatoria.

*Demostración.* La demostración se basa en el hecho de que  $X$  y  $g$  son funciones medibles, y que la composición de funciones medibles es una función medible. Es decir,  $g(X) : (\Omega, \mathcal{A}, P) \rightarrow (\mathbb{R}, \mathcal{B})$ , definida por  $g(X)(w) = g(X(w))$ , con  $w \in \Omega$ , es una función medible desde el espacio de probabilidad  $(\Omega, \mathcal{A}, P)$  hasta el espacio de medida  $(\mathbb{R}, \mathcal{B})$ . □

*Observación 1.39.* Sean  $X$  una variable aleatoria continua con función de densidad  $f_X$  y  $g$  una función integrable de Lebesgue. La esperanza de la variable aleatoria continua  $g(X)$  viene dada por

$$\mathbb{E}[g(X)] = \int_{-\infty}^{+\infty} g(x) f_X(x) dx.$$

Un resultado análogo se observaría para una variable aleatoria discreta, con la salvedad de que tendríamos la suma en lugar de la integral y la función de probabilidad en lugar de la función de densidad.

## 1. Probabilidad

Es conveniente remarcar la observación anterior, dado que es posible conocer la esperanza de la variable aleatoria  $g(X)$  conociendo la distribución de probabilidad de la variable  $X$ , sin la necesidad de conocer la propia distribución de  $g(X)$ . De igual manera, destacamos que estas propiedades de la esperanza se pueden generalizar para vectores aleatorios.

A partir de la definición de esperanza de una variable aleatoria se puede construir el concepto de varianza de la variable aleatoria. A modo intuitivo, la varianza es una medida estadística que nos ayudará a cuantificar la dispersión de un conjunto de datos en relación con su media (esperanza).

**Definición 1.40** (Varianza de una variable aleatoria). Sea  $X$  una variable aleatoria. Llamamos *varianza* de la variable aleatoria  $X$  ( $Var(X)$ ) al valor esperado dado por

$$Var(X) = \mathbb{E}[(X - \mathbb{E}[X])^2].$$

Además, denominaremos *desviación típica* ( $\sigma$ ) a  $\sqrt{Var(X)}$ .

*Observación 1.41.* La expresión de la varianza de una variable aleatoria puede expandirse de la siguiente manera:

$$\begin{aligned} Var(X) &= \mathbb{E}[(X - \mathbb{E}[X])^2] \\ &= \mathbb{E}[X^2 - 2X\mathbb{E}[X] + \mathbb{E}[X]^2] \\ &= \mathbb{E}[X^2] - 2\mathbb{E}[X]\mathbb{E}[X] + \mathbb{E}[X]^2 \\ &= \mathbb{E}[X^2] - \mathbb{E}[X]^2, \end{aligned}$$

donde se han aplicado algunas de las propiedades de la esperanza comentadas anteriormente.

## 1.3. Distribuciones de probabilidad

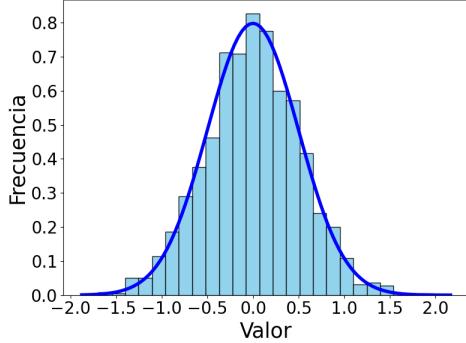
Las distribuciones de probabilidad son funciones que describen el comportamiento de una variable aleatoria, indicando la probabilidad de que esta tome ciertos valores. Estas distribuciones pueden ser, como se ha comentado en la sección anterior, discretas o continuas, dependiendo de la naturaleza de la variable aleatoria.

**Definición 1.42** (Moda). La *moda* de una distribución de probabilidad se refiere al dato (o datos) que más se repite en un conjunto de observaciones. En otras palabras, corresponde al valor (o valores) donde la función de probabilidad alcanza su máximo, o la función masa de probabilidad tiene la mayor frecuencia.

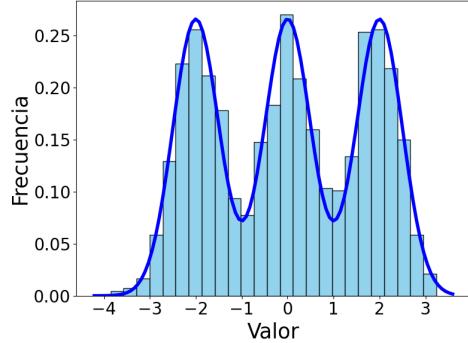
A su vez, diremos que una distribución es **unimodal** si posee un único valor de moda, **bimodal** si tiene dos, y **multimodal** si presenta varios (véase Figura 1.1).

### 1.3.1. Distribución Normal

La distribución normal, o distribución de Gauss, es la distribución de probabilidad más estudiada y utilizada en el ámbito de la inferencia estadística [Bry95], dadas sus propiedades matemáticas, como su simetría, la concentración de probabilidades alrededor de la media y su relación con otras distribuciones.



(a) Distribución unimodal.



(b) Distribución multimodal con tres modas.

Figura 1.1.: Ejemplos de distribuciones unimodal y multimodal. La distribución unimodal muestra un único máximo, mientras que la distribución multimodal presenta tres máximos, cada uno correspondiente a una moda.

**Definición 1.43** (Distribución normal). Sea  $X$  una variable aleatoria continua. Decimos que  $X$  sigue una *distribución normal* si su función de densidad  $f_X$  viene dada por:

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

donde  $\mu, \sigma^2 \in \mathbb{R}$  denotan, respectivamente, la esperanza y varianza de la variable aleatoria  $X$ .

Además, si  $X$  sigue una distribución normal, lo denotaremos como  $X \sim \mathcal{N}(\mu, \sigma^2)$ . De igual modo, si  $\mu = 0$  y  $\sigma^2 = 1$ , entonces diremos que la variable aleatoria  $X \sim \mathcal{N}(0, 1)$  sigue una *distribución normal estándar*.

**Proposición 1.44.** Si  $X \sim \mathcal{N}(\mu, \sigma^2)$ , entonces la variable aleatoria  $X$  satisface las siguientes propiedades:

- La distribución es simétrica respecto de su media  $\mu$ .
- La moda coincide con la media.
- $P[\mu - \sigma < X < \mu + \sigma] \approx 0,6826$ .
- $P[\mu - 2\sigma < X < \mu + 2\sigma] \approx 0,9544$ .
- $P[\mu - 3\sigma < X < \mu + 3\sigma] \approx 0,9974$ .
- Si  $a, b \in \mathbb{R}$ , entonces  $aX + b \sim \mathcal{N}(a\mu + b, a^2\sigma^2)$ .

Como consecuencia de la primera propiedad de la proposición anterior, se pueden relacionar todas las variables aleatorias normales con la distribución  $\mathcal{N}(0, 1)$ .

**Proposición 1.45.** Sea  $X \sim \mathcal{N}(\mu, \sigma^2)$ . Entonces,

$$Z = \frac{X - \mu}{\sigma}$$

es una variable aleatoria que sigue una distribución normal estándar  $Z \sim \mathcal{N}(0, 1)$ .

## 1. Probabilidad

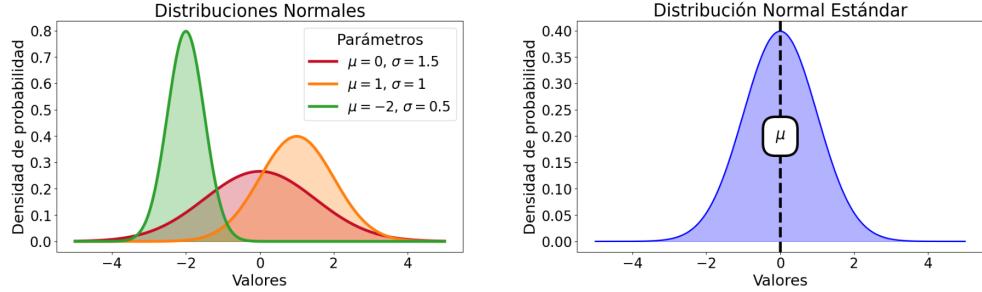


Figura 1.2.: Ejemplos de distribuciones normales. A la izquierda, observamos distintas distribuciones normales, donde podemos notar como el parámetro  $\mu$  determina el centro de la distribución y el parámetro  $\sigma$  controla la dispersión de los datos. A medida que el parámetro  $\sigma$  incrementa, la distribución se vuelve más dispersa y su altura máxima disminuye. A la derecha, observamos una distribución normal estándar, donde se puede apreciar con claridad que la distribución es simétrica con respecto a  $\mu$ .

### 1.3.2. Distribución de Rademacher

La distribución de Rademacher es una distribución de probabilidad discreta que se utiliza ampliamente en la teoría del aprendizaje estadístico, particularmente en el análisis de la capacidad de generalización de los modelos.

**Definición 1.46** (Distribución de Rademacher). Sea  $X$  una variable aleatoria discreta. Decimos que  $X$  sigue una *distribución de Rademacher* si toma los valores  $+1$  y  $-1$  con igual probabilidad, es decir,

$$P[X = 1] = P[X = -1] = \frac{1}{2}.$$

En tal caso, denotamos esta variable como  $X \sim \text{Rad}$ .

**Proposición 1.47.** Si  $X \sim \text{Rad}$ , entonces  $X$  es una variable aleatoria simétrica respecto al origen, con esperanza  $\mathbb{E}[X] = 0$  y varianza  $\text{Var}(X) = 1$ .

**Proposición 1.48.** Sean  $X_1, \dots, X_n$  variables independientes con  $X_i \sim \text{Rad}$ . Entonces, para cualquier conjunto de constantes  $a_1, \dots, a_n \in \mathbb{R}$ , la variable

$$S = \sum_{i=1}^n a_i X_i$$

es una combinación lineal de variables de Rademacher con esperanza cero y varianza  $\sum_{i=1}^n a_i^2$ .

## 2. Descomposición en Valores Singulares y Pseudoinversa de una Matriz

En este capítulo presentaremos dos conceptos fundamentales de cara al desarrollo del trabajo: la descomposición en valores singulares (SVD) y la pseudoinversa de una matriz. En cuanto a las referencias utilizadas a lo largo de este capítulo, debemos destacar [FIS14, STR23, POO11]. Estos libros nos han ayudado a presentar los conceptos básicos más importantes, así como las demostraciones de los resultados más relevantes del capítulo.

### 2.1. Vectores y matrices

En primer lugar, repasaremos la notación que utilizaremos para vectores y matrices, así como los principales tipos de matrices con las que trabajaremos. Además, recordaremos las condiciones bajo las cuales una matriz posee inversa.

Trabajaremos siempre con escalares reales y escribiremos los vectores de  $\mathbb{R}^n$  como vectores columna en lugar de vectores fila. Es decir, si  $v \in \mathbb{R}^n$ , entonces

$$v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = (v_1, v_2, \dots, v_n)^T, \quad \text{donde } v_i \in \mathbb{R} \quad \forall i \in \{1, \dots, n\}.$$

También se recuerdan los conceptos de norma y ortonormalidad de vectores, que serán de gran utilidad en el desarrollo de resultados posteriores. De esta manera, dados dos vectores  $u, v \in \mathbb{R}^n$ , denotaremos por  $u \cdot v$  su producto escalar usual y por  $\|u\|$  la norma euclídea del vector  $u$ . Así, diremos que dos vectores son ortogonales si su producto escalar es igual a cero.

Denotaremos por  $\mathcal{M}_{m \times n}(\mathbb{R})$  al espacio vectorial de las matrices de orden  $m$  por  $n$  con entradas en  $\mathbb{R}$ . A la matriz cuyas entradas son todas iguales a cero la llamaremos matriz cero y la denotaremos por  $O$ . Además, podemos formar una matriz a partir de vectores columna, utilizando la notación  $V = [v_1, v_2, \dots, v_n]$ , donde  $\{v_1, v_2, \dots, v_n\}$  son los vectores columna que componen la matriz  $V$ .

**Definición 2.1.** La matriz **traspuesta**  $A^T$  de una matriz  $A \in \mathcal{M}_{m \times n}(\mathbb{R})$  es la matriz de tamaño  $n \times m$  que se obtiene de la matriz  $A$  al intercambiar las filas por las columnas, es decir,  $(a^T)_{ij} = a_{ji}$ . Decimos que una matriz cuadrada  $A$  es **simétrica** si cumple  $A^T = A$  y, decimos que es **ortogonal** si sus columnas están formadas por vectores ortonormales dos a dos.

Denotamos la **delta de Kronecker**  $\delta_{ij}$  como la función dada por

$$\delta_{ij} = \begin{cases} 1 & \text{si } i = j \leq r, \\ 0 & \text{si } i \neq j. \end{cases}$$

De esta manera, la **matriz identidad** de tamaño  $n \times n$  ( $I_n$ ) viene dada por  $(I_n)_{ij} = \delta_{ij}$ .

## 2. Descomposición en Valores Singulares y Pseudoinversa de una Matriz

**Definición 2.2.** Sea  $A$  una matriz de tamaño  $m \times n$ . Se dice que  $A$  es una matriz **escalonada** si es la matriz  $O$ , o bien satisface las tres condiciones siguientes:

1. El primer elemento no nulo de cada fila, si existe, es un 1.
2. El primer 1 de la segunda y sucesivas filas está a la derecha del primer 1 de la fila anterior.
3. Si tiene filas nulas (compuestas únicamente por ceros), estas aparecen en la parte inferior de la matriz, justo debajo de las filas no nulas.

Además, las operaciones elementales que se pueden realizar a una matriz para obtener su forma escalonada son las siguientes:

- Intercambiar dos filas (columnas).
- Multiplicar una fila (columna) por un múltiplo distinto de cero.
- Sumar un múltiplo de una fila (columna) a otra fila (columna).

**Definición 2.3.** Sean  $A$  y  $B$  dos matrices de tamaño  $m \times n$ . Se dice que la matriz  $A$  es **equivalente** por filas a la matriz  $B$  (o simplemente equivalente) si  $B$  se obtiene de  $A$  por medio de la aplicación sucesiva de operaciones elementales.

**Definición 2.4.** Una matriz  $A$  de tamaño  $m \times n$  es **escalonada reducida** si es escalonada y, además, todo elemento en una columna que esté encima del primer uno de cualquier fila es cero.

**Ejemplo 2.5.** Para matrices cuadradas de tamaño 2, las posibles matrices escalonadas reducidas son las siguientes:

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & x \\ 0 & 0 \end{pmatrix} \quad y \quad \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},$$

donde  $x \in \mathbb{R}$  puede ser cualquier escalar.

Una vez introducidas las definiciones anteriores, disponemos de todas las herramientas necesarias para introducir el concepto de rango de una matriz.

**Definición 2.6** (Rango de una matriz). Sea  $A$  una matriz de tamaño  $m \times n$ . Se denomina **rango** de  $A$  ( $\text{rang}(A)$ ) al número de filas no nulas de la matriz en la forma escalonada reducida equivalente a  $A$ . De manera equivalente, el rango de una matriz se puede definir como la dimensión del espacio generado por sus vectores fila o columna. De esta manera, el rango de una matriz será igual al número máximo de vectores fila o columna que sean linealmente independientes entre sí.

El siguiente resultado nos recuerda cómo utilizar el rango para determinar si una matriz es invertible.

**Proposición 2.7.** Sea  $A$  una matriz cuadrada de tamaño  $n$ , entonces equivalen:

1.  $A$  es invertible.
2.  $A$  es equivalente a  $I_n$ .

3. El rango de  $A$  es  $n$ .
4.  $\det(A) \neq 0$ . Además, si  $A$  es invertible, entonces  $\det(A^{-1}) = \frac{1}{\det(A)}$ .

Finalmente, recordamos algunas de las propiedades de las matrices invertibles.

**Proposición 2.8.** Sean  $A$  y  $B$  matrices cuadradas e invertibles del mismo tamaño. Entonces se cumple:

1.  $A^{-1}$  es invertible y  $(A^{-1})^{-1} = A$  (la inversa de  $A^{-1}$  es la propia matriz  $A$ ).
2.  $A^T$  es invertible y  $(A^T)^{-1} = (A^{-1})^T$ .
3. La matriz  $AB$  es invertible y  $(AB)^{-1} = B^{-1}A^{-1}$ .

**Definición 2.9.** Sea  $A$  una matriz cuadrada de tamaño  $n$ . Un vector no nulo  $v \in \mathbb{R}^n$  se dice que es un **vector propio** (o autovector) de la matriz  $A$  si  $Av = \lambda v$  para algún escalar  $\lambda \in \mathbb{R}$ . Además, el escalar  $\lambda$  se llama **valor propio** (o autovalor) de la matriz  $A$  correspondiente al vector propio  $v$ . Recordemos que  $\lambda$  es un valor propio de  $A$  si, y solo si,  $\det(A - \lambda I_n) = 0$ .

Para cualquier matriz  $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ , la matriz  $A^T A$  es simétrica y, en consecuencia, es diagonalizable ortogonalmente (teorema espectral real [Blu21]). Se comprueba de manera sencilla que todos los valores propios de la matriz  $A^T A$  son no negativos.

## 2.2. SVD y pseudoinversa

Llegados a este punto, ya podemos presentar los principales contenidos sobre matrices abordados en este trabajo: la descomposición en valores singulares y la pseudoinversa. Cabe destacar que estos resultados se enuncian para matrices con escalares en el cuerpo  $\mathbb{R}$ , aunque su generalización a otros cuerpos es posible.

**Definición 2.10.** Si  $A$  es una matriz de tamaño  $m \times n$ , los **valores singulares** de  $A$  son las raíces cuadradas (positivas) de los valores propios de la matriz  $A^T A$ , y se denotan mediante  $\sigma_1, \sigma_2, \dots, \sigma_n$ . Además, es convencional ordenar los valores singulares de manera que  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n$ .

El siguiente resultado nos indica que toda matriz, independientemente de su estructura, puede ser factorizada como producto de tres matrices, dos de las cuales serán ortogonales.

**Teorema 2.11** (Descomposición en valores singulares). *Sea  $A \in \mathcal{M}_{m \times n}(\mathbb{R})$  una matriz cuyo rango es  $r$  y con valores singulares positivos  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$ , y sea  $\Sigma$  la matriz de tamaño  $m \times n$  definida por*

$$\Sigma_{ij} = \begin{cases} \sigma_i & \text{si } i = j \leq r, \\ 0 & \text{en otro caso.} \end{cases}$$

*Entonces existen matrices ortogonales  $U$  y  $V$  de tamaño  $m \times m$  y  $n \times n$ , respectivamente, de manera que*

$$A = U\Sigma V^T.$$

*A esta factorización la llamaremos descomposición en valores singulares (SVD) de  $A$ .*

## 2. Descomposición en Valores Singulares y Pseudoinversa de una Matriz

*Demostración.* La demostración se fundamenta en la construcción directa de las matrices  $V$  y  $U$ , verificando posteriormente que se satisface el resultado buscado.

Para construir la matriz ortogonal  $V$ , basta tomar una base ortonormal  $\{v_1, v_2, \dots, v_n\}$  de  $\mathbb{R}^n$  formada por vectores asociados a los valores propios  $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$  de la matriz simétrica y cuadrada  $A^T A$  de tamaño  $n$ . Entonces, se tiene que  $V = [v_1, v_2, \dots, v_n]$  es una matriz ortogonal y cuadrada de tamaño  $n$ .

Para construir la matriz ortogonal  $U$ , primero notemos que  $\{Av_1, Av_2, \dots, Av_n\}$  es un conjunto ortogonal de vectores de  $\mathbb{R}^m$ . En efecto, para  $i \neq j$ , se cumple

$$(Av_i) \cdot (Av_j) = (Av_i)^T Av_j = v_i^T A^T Av_j = v_i^T \lambda_j v_j = \lambda_j(v_i \cdot v_j) = 0,$$

dado que los vectores propios  $v_i$  son ortogonales. Recordamos ahora que los valores singulares satisfacen  $\sigma_i = \|Av_i\|$  y que los primeros  $r$  valores singulares son distintos de cero. Por tanto, podemos normalizar  $Av_1, \dots, Av_r$  de la siguiente forma:

$$u_i = \frac{1}{\sigma_i} Av_i, \quad \text{para } i = \{1, \dots, r\}. \quad (2.1)$$

Esto garantiza que  $\{u_1, \dots, u_r\}$  es un conjunto ortonormal de  $\mathbb{R}^m$ , pero si  $r < m$  no será una base para  $\mathbb{R}^m$ . En este caso, se extiende el conjunto  $\{u_1, \dots, u_r\}$  a una base ortonormal  $\{u_1, \dots, u_m\}$  para  $\mathbb{R}^m$ . En consecuencia, la matriz  $U$  es ortogonal. Ahora, falta comprobar que, con la construcción realizada, se satisface el resultado  $A = U\Sigma V^T$ . Dado que  $V^T = V^{-1}$  (al ser la matriz  $V$  ortogonal), esto equivale a demostrar que  $AV = U\Sigma$ .

En primer lugar, sabemos, a partir de la ecuación (2.1), que  $Av_i = \sigma_i u_i$  para  $i = \{1, \dots, r\}$  y que  $\|Av_i\| = \sigma_i = 0$  para  $i = \{r+1, \dots, n\}$ . En consecuencia,  $Av_i = 0$  para  $i = \{r+1, \dots, n\}$ . Por tanto,

$$\begin{aligned} AV &= A \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{bmatrix} \\ &= \begin{bmatrix} A\mathbf{v}_1 & \cdots & A\mathbf{v}_n \end{bmatrix} \\ &= \begin{bmatrix} A\mathbf{v}_1 & \cdots & A\mathbf{v}_r & 0 & \cdots & 0 \end{bmatrix} \\ &= \begin{bmatrix} \sigma_1 \mathbf{u}_1 & \cdots & \sigma_r \mathbf{u}_r & 0 & \cdots & 0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_m \end{bmatrix} \begin{bmatrix} \sigma_1 & \cdots & 0 & O \\ \vdots & \ddots & \vdots & O \\ 0 & \cdots & \sigma_r & O \\ O & O & O & O \end{bmatrix} \\ &= U\Sigma, \end{aligned}$$

quedando probada la igualdad  $AV = U\Sigma$ . □

El siguiente resultado generaliza el concepto de matriz inversa cuando la matriz no es cuadrada.

**Definición 2.12** (Pseudoinversa). Sea  $A \in \mathcal{M}_{m \times n}(\mathbb{R})$  con  $m > n$  y cuyas columnas son linealmente independientes. Se define la **pseudoinversa** (o inversa de Moore-Penrose) de la matriz  $A$  como la matriz  $A^\dagger$  dada por

$$A^\dagger = (A^T A)^{-1} A^T,$$

donde se puede comprobar que  $A^\dagger \in \mathcal{M}_{n \times m}(\mathbb{R})$ .

No obstante, dado que toda matriz se puede factorizar en su descomposición en valores singulares, podemos definir la pseudoinversa de una matriz a partir de dicha factorización.

**Definición 2.13.** Sea  $A$  una matriz de tamaño  $m \times n$  de rango  $r$  con descomposición en valores singulares  $A = U\Sigma V^T$  y con valores singulares distintos de cero  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$ . Sea  $\Sigma^\dagger$  la matriz de tamaño  $n \times m$  definida por

$$\Sigma_{ij}^\dagger = \begin{cases} \frac{1}{\sigma_i} & \text{si } i = j \leq r, \\ 0 & \text{en otro caso.} \end{cases}$$

Entonces la factorización  $A^\dagger = V\Sigma^\dagger U^T$  es una descomposición en valores singulares de  $A^\dagger$ , donde  $\Sigma^\dagger$  es la pseudoinversa de  $\Sigma$ . Además,  $A^\dagger$  es la pseudoinversa de  $A$ .

Presentamos una forma análoga de definir la pseudoinversa de una matriz, basándose en las propiedades que debe cumplir la pseudoinversa, que serán de gran utilidad en el desarrollo del trabajo.

**Definición 2.14** (Condiciones de Moore-Penrose). Sea  $A \in \mathcal{M}_{m \times n}(\mathbb{R})$  la pseudoinversa de  $A$ ,  $A^\dagger \in \mathcal{M}_{n \times m}(\mathbb{R})$ , es la única matriz que satisface las siguientes propiedades, conocidas como las **condiciones de Moore-Penrose**:

1.  $AA^\dagger A = A$ .
2.  $A^\dagger AA^\dagger = A^\dagger$ .
3.  $(AA^\dagger)^T$  y  $(A^\dagger A)^T$  son simétricas.

En particular, se tiene que  $AA^\dagger$  y  $A^\dagger A$  son proyecciones ortogonales sobre  $Im(A)$  y  $Im(A)^T$  respectivamente. Además, si  $A$  es de rango completo, es decir,  $rango(A) = r = \min\{m, n\}$ , entonces  $A^\dagger$  puede expresarse de forma sencilla como sigue:

- Si  $r = m = n$ , entonces la matriz  $A$  es invertible y  $A^\dagger = A^{-1}$ .
- Si  $r = m < n$ , entonces  $A$  tiene filas linealmente independientes (la aplicación lineal asociada a  $A$  es sobreyectiva y  $AA^T$  es invertible) y  $A^\dagger = A^T(AA^T)^{-1}$ .
- Si  $r = n < m$ , entonces  $A$  tiene columnas linealmente independientes (la aplicación lineal asociada a  $A$  es inyectiva y  $A^TA$  es invertible) y  $A^\dagger = (A^TA)^{-1}A^T$ .

A continuación, se presentan dos propiedades de la pseudoinversa que son fundamentales para comprender la estructura de la solución de sistemas lineales.

**Lema 2.15.** Para una matriz  $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ , se verifica:

1.  $Im(I - A^\dagger A) = \ker(A)$ .
2.  $\ker(A^\dagger) = \ker(A^T)$ .

Donde  $Im(A)$  y  $\ker(A)$  denotan, respectivamente, la imagen y el núcleo de la aplicación lineal asociada a la matriz  $A$ .

## 2. Descomposición en Valores Singulares y Pseudoinversa de una Matriz

Finalmente, la solución de norma mínima para un problema de mínimos cuadrados puede definirse mediante la pseudoinversa de una matriz, como se establece en el siguiente teorema. Más adelante, este resultado será de gran utilidad para obtener la mejor aproximación en problemas de regresión.

**Teorema 2.16.** *El problema de mínimos cuadrados  $Ax = \mathbf{b}$ , con  $A \in \mathcal{M}_{m \times n}(\mathbb{R})$ ,  $x \in \mathbb{R}^n$  y  $\mathbf{b} \in \mathbb{R}^m$ , tiene una solución única  $\bar{x}$  de mínimos cuadrados con norma mínima dada por*

$$\bar{x} = A^\dagger \mathbf{b}.$$

*Demostración.* Sea  $A \in \mathcal{M}_{m \times n}(\mathbb{R})$  con  $\text{rang}(A) = r$  y sea  $U\Sigma V^T$  su descomposición en valores singulares. De este modo, se tiene que  $A^\dagger = V\Sigma^\dagger U^T$ . Sean  $\mathbf{y} = V^T x$  y  $\mathbf{c} = U^T \mathbf{b}$ , expresados de la siguiente forma

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix},$$

donde  $\mathbf{y}_1, \mathbf{c}_1 \in \mathbb{R}^r$ .

Se busca minimizar  $\|\mathbf{b} - Ax\|$  o, de manera equivalente,  $\|\mathbf{b} - Ax\|^2$ . Usando que  $U^T$  es ortogonal (dados que  $U$  es ortogonal), se tiene

$$\begin{aligned} \|\mathbf{b} - Ax\|^2 &= \|U^T(\mathbf{b} - Ax)\|^2 = \|U^T(\mathbf{b} - U\Sigma V^T x)\|^2 = \|U^T \mathbf{b} - U^T U \Sigma V^T x\|^2 \\ &= \|\mathbf{c} - \Sigma \mathbf{y}\|^2 = \left\| \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix} - \begin{bmatrix} D & O \\ O & O \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} \right\|^2 = \left\| \begin{bmatrix} \mathbf{c}_1 - D\mathbf{y}_1 \\ \mathbf{c}_2 \end{bmatrix} \right\|^2. \end{aligned}$$

Dado que sólo disponemos de control sobre  $\mathbf{y}_1$ , el valor mínimo ocurre cuando  $\mathbf{c}_1 - D\mathbf{y}_1 = 0$  o, de manera equivalente, cuando  $\mathbf{y}_1 = D^{-1}\mathbf{c}_1$ . De modo que todas las soluciones  $x$  de mínimos cuadrados son de la forma

$$x = V\mathbf{y} = V \begin{bmatrix} D^{-1}\mathbf{c}_1 \\ \mathbf{y}_2 \end{bmatrix}.$$

Definimos  $\bar{x} = V\bar{\mathbf{y}} = V \begin{bmatrix} D^{-1}\mathbf{c}_1 \\ 0 \end{bmatrix}$  y afirmamos que  $\bar{x}$  es la solución de mínimos cuadrados de norma mínima. Para demostrarlo, supongamos que  $x' = V\mathbf{y}' = V \begin{bmatrix} D^{-1}\mathbf{c}_1 \\ \mathbf{y}_2 \end{bmatrix}$  es otra solución diferente al problema de mínimos cuadrados (por tanto,  $\mathbf{y}_2 \neq 0$ ). Entonces, se verifica

$$\|\bar{x}\| = \|V\bar{\mathbf{y}}\| = \|\bar{\mathbf{y}}\| < \|\mathbf{y}'\| = \|V\mathbf{y}'\| = \|x'\|,$$

como se quería probar. Por último, falta demostrar que  $\bar{x}$  es igual a  $A^\dagger \mathbf{b}$ . Para ello, basta calcular

$$\bar{x} = V\bar{\mathbf{y}} = V \begin{bmatrix} D^{-1}\mathbf{c}_1 \\ 0 \end{bmatrix} = V \begin{bmatrix} D^{-1} & O \\ O & O \end{bmatrix} \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix} = V\Sigma^\dagger \mathbf{c} = V\Sigma^\dagger U^T \mathbf{b} = A^\dagger \mathbf{b}.$$

□

## 3. Aprendizaje Automático y Aprendizaje Profundo

### 3.1. Fundamentos

El aprendizaje automático o *machine learning* (ML) [Biso6, Mur22, Mur23] es una rama de la inteligencia artificial que, mediante el uso de datos y algoritmos de aprendizaje, proporciona a las máquinas la capacidad de aprender de manera automática de los datos. En otras palabras, el aprendizaje automático se encarga de utilizar un conjunto de observaciones para descubrir un proceso subyacente en los mismos [AMMIL12], con el objetivo de imitar el comportamiento humano, identificando patrones y relaciones en los datos.

En términos generales, el aprendizaje automático se divide en tres tipos: el aprendizaje supervisado, que actúa sobre datos etiquetados (cada ejemplo de entrada se encuentra asociado a una salida conocida), el aprendizaje no supervisado, que trabaja sobre datos no etiquetados, donde es el propio sistema el que debe ser capaz de reconocer los patrones subyacentes de los datos mediante el uso exclusivo de los ejemplos de entrada, y el aprendizaje por refuerzo, que actúa en un entorno de ensayo-error, donde el modelo aprende a través de recompensas y penalizaciones que se le otorgan a medida que realiza las acciones. Para nuestro trabajo, nos limitaremos a trabajar con el **aprendizaje supervisado**.

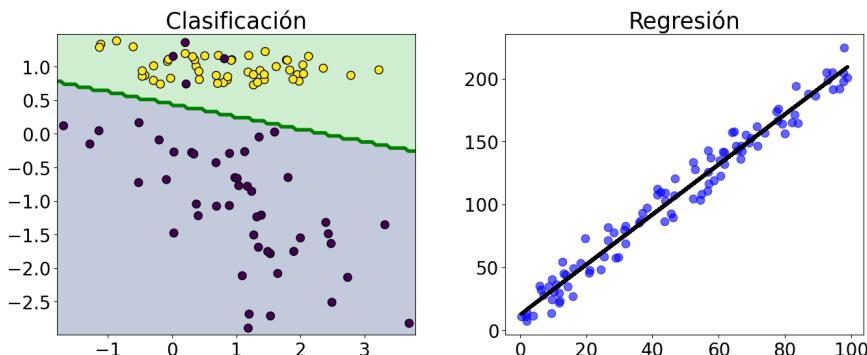


Figura 3.1.: Ejemplos de problemas de clasificación y regresión. A la izquierda, se muestra un problema de clasificación binario, donde se busca encontrar el hiperplano (línea verde) que separe ambos conjuntos de datos etiquetados. A la derecha, se muestra un problema de regresión, donde se busca encontrar la mejor aproximación (línea negra) al conjunto de datos.

Dentro del marco del aprendizaje supervisado, donde el principal objetivo del modelo es aprender a predecir la salida correcta para nuevos ejemplos no conocidos, basándose en las relaciones y patrones extraídos al trabajar con los datos etiquetados, podemos dividir los problemas en dos categorías principales: problemas de clasificación, en los que se asigna una salida o clase (discreta) a cada entrada, y problemas de regresión, en los que se predice un valor continuo para cada entrada (véase Figura 3.1). De cara al desarrollo de nuestro trabajo, abordaremos ambos tipos de problemas. En particular, en los problemas de clasificación, nos

### 3. Aprendizaje Automático y Aprendizaje Profundo

centraremos en la clasificación de imágenes.

El aprendizaje profundo o *deep learning* (*DL*) [BB24, Pri23, LBH15] representa un área del aprendizaje automático que utiliza redes neuronales artificiales, inspiradas en la estructura y funcionamiento del cerebro humano, con múltiples capas, conocidas como redes neuronales profundas [GBC16, Sch15], con el propósito de identificar y modelar patrones complejos y extraer representaciones jerárquicas en grandes volúmenes de datos.

## 3.2. Redes neuronales artificiales

Una red neuronal artificial o *artificial neural network* (*ANN*) [Bis95, Rip96] es un modelo de aprendizaje automático que toma decisiones de manera similar al funcionamiento del cerebro humano, a partir de las interconexiones que presentan las neuronas biológicas, que se organizan en diferentes capas interconectadas. Estas conexiones simulan las interacciones entre las neuronas biológicas, permitiendo que la red procese información y aprenda de manera similar al propio cerebro humano.

De manera similar al cerebro humano, una red neuronal artificial está formada por neuronas artificiales (véase Figura 3.2), también llamadas unidades. Estas unidades se agrupan en diferentes capas formando la arquitectura global de la red neuronal. Cada capa puede contener un número variable de unidades, lo que permite adaptar la red a la complejidad del problema a resolver.

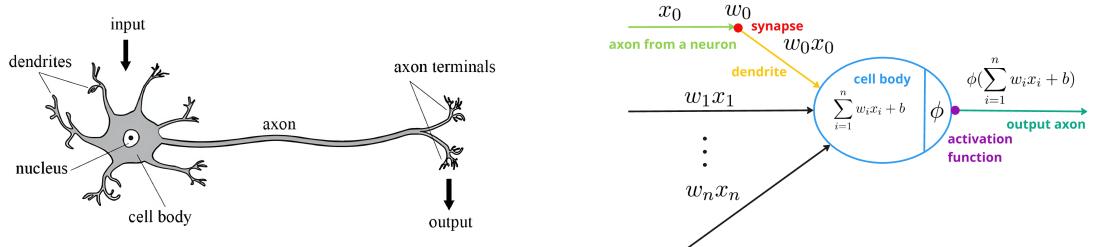


Figura 3.2.: Ejemplos de una neurona biológica [NGLK18] (izquierda) y una neurona artificial (derecha) basada en [LJY24].

A alto nivel, el funcionamiento de una red neuronal artificial se divide en, al menos, tres capas principales, constituidas por una capa de entrada, una capa oculta y una capa de salida. En la primera capa o capa de entrada, la información del mundo exterior entra en la red neuronal. Dicha información es procesada y propagada al resto de capas mediante un proceso conocido como **propagación hacia delante** o *forward propagation*, permitiendo que la información fluya desde la capa de entrada hacia las capas sucesivas.

En la capa oculta o capa de procesamiento, las unidades reciben las salidas de la capa anterior y se encargan de procesar la información mediante conexiones ponderadas (llamadas pesos) y funciones de activación (encargadas de introducir no linealidad en el modelo, permitiendo que la red aprenda y represente relaciones complejas entre las entradas y las salidas), extrayendo características y patrones relevantes de los datos. Los pesos obtenidos controlan la influencia que cada neurona de la capa anterior tiene sobre la neurona actual.

Las redes neuronales artificiales pueden tener una gran cantidad de capas ocultas, lo que permite un procesamiento más profundo y detallado de la información. Cada capa oculta analiza la salida de la capa anterior, la procesa aún más y la pasa a la siguiente capa, de

modo que, a medida que se avanza por la red, se generan representaciones internas cada vez más abstractas de la entrada original.

Finalmente, en la última capa o capa de salida, la red produce el resultado final en función del problema en cuestión y de la predicción calculada, haciendo uso de los pesos ajustados en las capas ocultas. La naturaleza de la salida varía según el tipo de tarea que se realice: en un problema de clasificación, la salida es un valor discreto que indica la clase a la que pertenece la entrada, mientras que, en regresión, la salida es un valor continuo que representa una predicción numérica. Por tanto, esta capa es la que traduce la información procesada por la red en un resultado interpretable y acorde con el objetivo del problema.

Por consiguiente, el proceso de entrenamiento de una red neuronal artificial es un proceso iterativo en el que la red ajusta sus pesos para aprender a realizar tareas específicas. Para llevar a cabo este proceso, es fundamental un conjunto de datos o ejemplos de entrenamiento que sean lo suficientemente representativos, ya que de este conjunto se extraerán los patrones relevantes que la red aprenderá.

### 3.2.1. Redes neuronales convolucionales

Las redes neuronales convolucionales o *convolutional neural networks* (CNN) [LBD<sup>+</sup>89, LBBH98] son un tipo especial de red neuronal artificial que se utiliza principalmente en procesamiento de imágenes, reconocimiento visual y tareas relacionadas con datos que tienen una estructura de rejilla (matriz multidimensional), como imágenes, vídeos o señales de audio.

Una de las características más destacadas de las CNN es su capacidad para realizar la extracción automática y jerárquica de características de los datos de entrada, lo que las hace especialmente poderosas para tareas que requieren reconocer patrones complejos en los datos. Esta capacidad es una de las principales razones por las que las CNN son tan eficaces, ya que permiten a la red identificar patrones relevantes sin necesidad de intervención humana para diseñar características específicas, lo que las vuelve especialmente interesantes en áreas como la visión por computador, donde han impulsado avances significativos en aplicaciones como el reconocimiento de imágenes, la segmentación semántica y la detección de objetos, entre otras.

Las CNN incluyen varias capas especializadas que las distinguen de las redes neuronales artificiales tradicionales: las capas de convolución, encargadas de la extracción de características de la entrada; las capas de agrupación o *pooling*, responsables de la reducción de la dimensionalidad sin perder información clave, y las capas totalmente conectadas o *fully connected*, que combinan las características extraídas para realizar la predicción final (véase Figura 3.3). Estas capas operan de manera conjunta, transformando la entrada y refinando progresivamente las características conforme se avanza por la red.

A continuación, se describen las principales capas que conforman una red neuronal convolucional. Se incluyen tanto las capas exclusivas de este tipo de red como aquellas que comparte con las redes neuronales tradicionales.

#### 3.2.1.1. Capa de convolución

La capa convolucional es un componente fundamental y exclusivo de las CNN, diseñada para extraer características locales de datos estructurados en forma de matrices multidimensionales. Su funcionamiento se basa en utilizar matrices de valores, conocidas como filtros o *kernels*, que se deslizan sobre la entrada aplicando la operación matemática de convolución.

### 3. Aprendizaje Automático y Aprendizaje Profundo

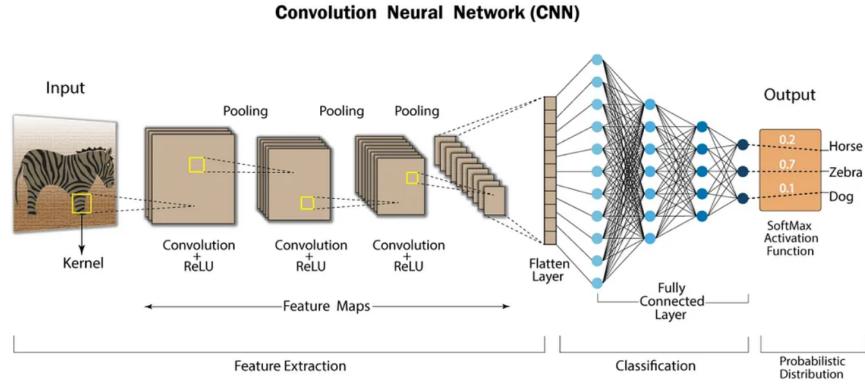


Figura 3.3.: Ejemplo de CNN utilizada para clasificación de imágenes [Swa20]. La entrada a la red es una imagen en formato RGB. La extracción de características se realiza mediante varias capas convolucionales, seguidas de funciones de activación y capas de *pooling*, las cuales ayudan a reducir la dimensionalidad. Posteriormente, las características extraídas se apllanan (*flattening*) y se envían a las capas totalmente conectadas. Finalmente, la salida pasa por una función de activación, en este caso *softmax*, que genera una distribución de probabilidad sobre las posibles clases de salida.

La convolución es una operación lineal que consiste en desplazar un filtro sobre la entrada, realizando en cada posición una multiplicación elemento a elemento entre los valores del filtro y los de la entrada (diferente de una multiplicación matricial convencional). Posteriormente, se suman estos productos para obtener un único valor de salida. Este proceso se repite hasta deslizar el filtro a lo largo de toda la entrada, obteniendo una nueva matriz denominada **mapa de características**.

Los valores de los filtros actúan como los pesos que la propia red aprende y optimiza de forma iterativa para maximizar la extracción de características relevantes de la entrada. Además, el número de filtros aplicados sobre cada entrada influye directamente en el mapa de características de salida. Así, a mayor cantidad de filtros, la profundidad del mapa de características resultante también será mayor.

Como se puede observar en la Figura 3.4, la propia naturaleza de la operación de convolución modifica la dimensionalidad de la entrada. Sin embargo, esto no siempre es deseable, ya que en determinadas ocasiones preferiremos mantener la dimensión original de la entrada. Para solucionar este problema, se introduce el concepto de **relleno** o *padding*, consistente en agregar información adicional alrededor de la entrada, con el fin de preservar su dimensionalidad.

Por otra parte, la elección de la siguiente zona sobre la que se realizará la convolución viene determinada por el tamaño de paso o *stride*. Generalmente, se utiliza un *stride* de 1, lo que significa que desplazamos el filtro de manera adyacente una posición en cada paso. Sin embargo, también es posible reducir la dimensionalidad de la entrada modificando el *stride*, como puede observarse en la Figura 3.4, donde, a pesar de utilizar un relleno de una posición para mantener la dimensionalidad, el tamaño del mapa de activación resultante es inferior al tamaño original de la entrada, pues se está utilizando un *stride* de 2.

En conclusión, el objetivo principal de la capa de convolución es extraer características relevantes de la entrada. Para ello, se utilizan filtros cuyos pesos son aprendidos y optimizados

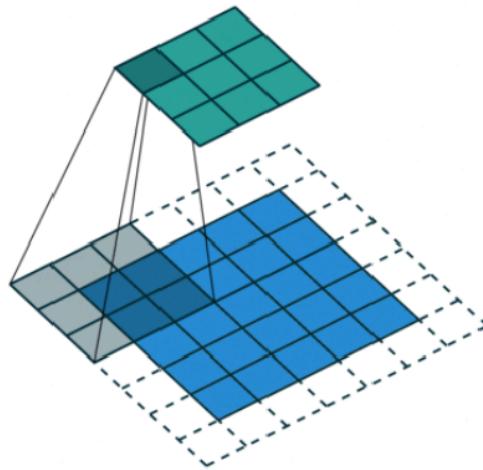


Figura 3.4.: Ejemplo de convolución con *padding* [Sah18]. El color azul hace referencia a la entrada, mientras que el color verde denota el resultado de la convolución. En particular, el color verde oscuro destaca el resultado de la convolución que se realiza sobre la zona grisácea de la imagen, utilizando, en este caso, un filtro de tamaño  $3 \times 3$ . Por otra parte, las cuadrículas punteadas hacen referencia al *padding* agregado.

por la propia red. En las primeras capas convolucionales, dado que el tamaño de la entrada es mayor, se detectan principalmente características de bajo nivel, como bordes y texturas. A medida que la información avanza por la red, las capas posteriores capturan patrones más complejos y abstractos. De este modo, mediante la combinación de múltiples capas convolucionales, la red logra desarrollar una representación jerárquica de la entrada.

### 3.2.1.2. Capa de *pooling*

Las capas de *pooling* son otro componente esencial y exclusivo en las redes neuronales convolucionales, ya que su función principal es reducir la dimensionalidad de las representaciones producidas por las capas de convolución, simplificando los mapas de características mientras se preservan las características más relevantes, lo que conlleva una reducción en la cantidad de parámetros y en la complejidad computacional de la red.

El *pooling* es una operación que toma un conjunto de valores de un mapa de características y lo reduce a un solo valor, con el propósito de submuestrear la información, introduciendo cierta invarianza espacial frente a pequeñas variaciones espaciales de la entrada. Esto permite detectar patrones aunque se encuentren ligeramente desplazados en la imagen, aumentando la robustez de la red.

El tipo de *pooling* más comúnmente utilizado es el denominado *max pooling* (véase Figura 3.4), que selecciona el valor máximo de un conjunto de valores dentro de una región del mapa de características. Otras alternativas incluyen el *average pooling*, que selecciona el valor promedio del conjunto de valores de la región del mapa de características utilizada y el *global pooling*, que reduce el mapa de características a un único valor. En nuestro proyecto se hará uso del primero de estos, integrado en algunas de las arquitecturas que utilizamos para los

### 3. Aprendizaje Automático y Aprendizaje Profundo

experimentos.

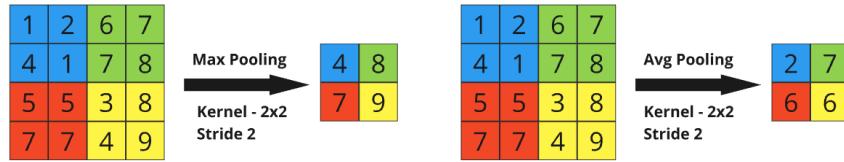


Figura 3.5.: Ejemplos de *pooling* utilizados en CNN. A la izquierda, se muestra el *max pooling*, donde se selecciona el valor máximo de una determinada región (en este caso  $2 \times 2$ ). A la derecha, se visualiza el *average pooling*, donde se selecciona el valor promedio de la región influida por el filtro.

#### 3.2.1.3. Capa totalmente conectada

Las capas totalmente conectadas o densas (*fully connected*) son un componente fundamental en las redes neuronales, presentes tanto en las tradicionales, formando la arquitectura básica de una ANN, como en las redes convolucionales. Estas capas se encuentran normalmente en las etapas finales de la red, encargándose de realizar la interpretación final de las características extraídas por las capas anteriores (véase Figura 3.3).

Antes de entrar a una capa densa, la salida de las capas anteriores debe aplanarse (*flattening*) en un vector unidimensional puesto que, generalmente, la salida de las capas anteriores tendrán forma matricial o tensorial, lo que impide que puedan ser procesadas directamente por las capas densas. De esta manera, el número de unidades de la primera capa densa se corresponde con el número de componentes del vector unidimensional.

Estas capas combinan y procesan las características extraídas de las capas anteriores, y tienen la posibilidad de capturar relaciones globales al conectar cada unidad de una capa con todas las unidades de la siguiente capa por medio de conexiones con pesos entrenables, lo que produce que la mayor parte de los pesos entrenables de una red suelan concentrarse en estas capas, debido al elevado número de conexiones que presentan.

#### 3.2.1.4. Capa de activación

Las capas o **funciones de activación** son las responsables de introducir no linealidad en el modelo, transformando la combinación lineal de las entradas mediante una función matemática. Esta transformación es esencial para que la red pueda aprender y representar patrones complejos en los datos.

Sin funciones de activación, una red neuronal se reduciría simplemente a una combinación lineal de las entradas, sin importar cuántas capas tuviera. Como menciona Bishop en su libro [Biso6]: “*Si se considera que las funciones de activación de todas las unidades ocultas de una red son lineales, entonces para cualquier red de este tipo siempre podemos encontrar una red equivalente sin unidades ocultas*”, limitando la capacidad de la red para modelar relaciones complejas.

Por tanto, las capas de activación se colocan después de cada capa lineal (como las capas convolucionales o las capas densas) en una red neuronal, con el objetivo de permitir que dicha capa pueda aprender también relaciones no lineales. A lo largo de este trabajo, se utilizarán algunas de las funciones de activación más comunes y ampliamente empleadas en el campo del aprendizaje profundo, entre las que se incluyen:

- **ReLU (Rectified Linear Unit):** Es una de las funciones de activación más utilizadas, especialmente en las capas ocultas de las redes neuronales profundas, debido a su simplicidad computacional, lo que permite acelerar el proceso de entrenamiento. Su expresión matemática es la siguiente:

$$\text{ReLU}(x) = \max(0, x),$$

donde  $x$  representa la entrada a la función, que corresponde con la salida lineal de la capa anterior de la red neuronal. Sin embargo, esta función de activación puede provocar el problema de “neuronas muertas”, donde algunas unidades dejan de activarse permanentemente, cuando su entrada es negativa o 0 (véase Figura 3.6).

- **Softmax:** Se utiliza principalmente en la capa de salida para tareas de clasificación multiclase. Convierte un vector de valores reales en un vector de probabilidades que suman 1, facilitando la interpretación de las salidas como probabilidades de pertenencia a cada clase. Su expresión matemática es la siguiente:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}},$$

donde  $x_i$  representa la entrada correspondiente a la clase  $i$ -ésima antes de aplicar la activación y  $K$  el número de clases. Esta función de activación es una generalización de la función sigmoide para clasificación multiclase. Es por esto que, para el caso de  $K = 2$  (clasificación binaria), esta función se reduce a la función sigmoide.

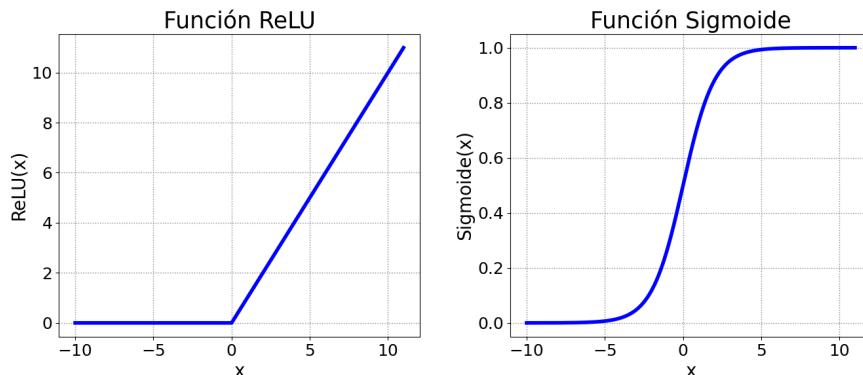


Figura 3.6.: Ejemplos de funciones de activación utilizadas en CNN. A la izquierda, aparece la función ReLU, donde se observa como deja invariantes los valores positivos, estableciendo a cero los valores negativos. A la derecha, se presenta la función sigmoide, que mapea los valores de entrada a un rango entre 0 y 1.

## 4. El Dilema Clásico del Aprendizaje

En este capítulo, como preámbulo antes de adentrarnos en el *Deep Double Descent*, presentaremos algunos conceptos básicos de la sabiduría clásica del aprendizaje automático, que nos ayudarán a entenderlo de manera más precisa. Las fuentes principales utilizadas a lo largo de este capítulo son extractos de [[AMMIL12](#), [Bis06](#)].

### 4.1. Concepto de aprendizaje

El aprendizaje, dentro del marco del aprendizaje automático, puede considerarse como un proceso en el que se busca encontrar una función  $g$  que aproxime lo máximo posible a la función objetivo  $f$ , que describe las relaciones y patrones subyacentes entre las entradas y salidas de los datos. Dado que la función objetivo es siempre desconocida, pues, en otro caso, no habría nada que aprender, serán los propios datos etiquetados los que nos ayuden, mediante el entrenamiento de modelos, a obtener una función aproximadora de dicha función objetivo.

En nuestro caso, de cara a trabajar con el aprendizaje supervisado, consideraremos los siguientes componentes del mismo:

- Espacio muestral  $\mathcal{X}$ : representa el conjunto de todas las posibles entradas que el modelo puede recibir, tomadas de manera independiente siguiendo alguna (sin restricción) distribución de probabilidad  $P$  en  $\mathcal{X}$ .
- Conjunto  $\mathcal{Y}$ : compuesto por todas las posibles salidas (etiquetas) que el modelo debe predecir.
- Función  $f : \mathcal{X} \rightarrow \mathcal{Y}$ , que representa la función objetivo (desconocida) que asigna cada entrada  $x \in \mathcal{X}$  a una salida  $y \in \mathcal{Y}$ .
- Un conjunto de datos de entrenamiento  $\mathcal{D}$ , formado por pares  $(x, y)$  con  $x \in \mathcal{X}$  e  $y \in \mathcal{Y}$ , donde  $f(x) = y$ .
- Un conjunto de hipótesis  $\mathcal{H}$ , donde se encontrarán todas las funciones candidatas que el modelo puede aprender para aproximar la función objetivo. Es decir,  $\mathcal{H} = \{h : X \rightarrow Y / X \subseteq \mathcal{X}, Y \subseteq \mathcal{Y}\}$ .
- Un algoritmo de aprendizaje  $\mathcal{A}$ , que es el encargado de elegir una función candidata  $h \in \mathcal{H}$  que aproxime a la función objetivo  $f$ .

De este modo, el modelo de aprendizaje automático, por medio del algoritmo de aprendizaje, será el encargado de seleccionar la función candidata  $g \in \mathcal{H}$  que mejor aproxime a la función objetivo  $f$ , utilizando el conjunto de datos de entrenamiento disponible, con el propósito de que la función candidata siga replicando a la función objetivo ante nuevos datos no disponibles (véase Figura 4.1). Este es, en esencia, el objetivo final del aprendizaje.

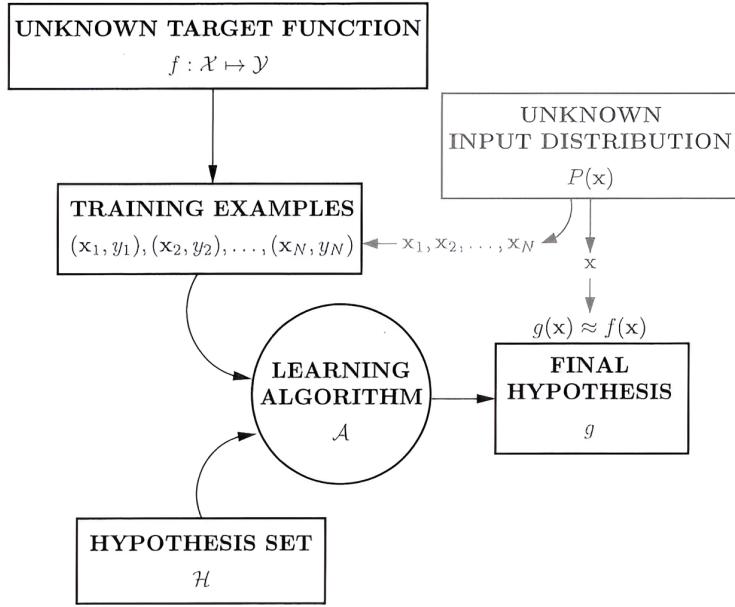


Figura 4.1.: Diagrama representando el concepto clásico de aprendizaje [AMMIL12]. El algoritmo de aprendizaje (*learning algorithm*), utilizando los ejemplos de entrenamiento (*training examples*) muestreados de una distribución de probabilidad desconocida (*unknown input distribution*), buscará en el conjunto de hipótesis (*hypothesis set*) la mejor aproximación (*final hypothesis*) a la función objetivo desconocida (*unknown target function*) que se desea aprender.

### 4.1.1. Descenso de gradiente y aprendizaje

El descenso de gradiente o *gradient descent (GD)* es la columna vertebral del aprendizaje en redes neuronales y, en general, sienta las bases para las técnicas de aprendizaje automático y aprendizaje profundo. Se trata de un algoritmo de optimización sin restricciones cuyo objetivo principal es minimizar la función de pérdida o error del modelo. Dicha función de pérdida mide cuán lejos están las predicciones realizadas por el modelo de los valores reales, y minimizarla conllevará asociada una mejora en su precisión<sup>1</sup>.

La idea subyacente del descenso de gradiente se basa en calcular de forma iterativa el gradiente, es decir, la derivada parcial de la función de pérdida con respecto a los parámetros. A partir de este gradiente, nos desplazamos en la dirección opuesta al mismo (véase ecuación (4.1)), ya que esta zona indica la dirección del descenso más pronunciado en la función de pérdida. De este modo, se garantiza que los parámetros se ajusten para minimizar progresivamente la pérdida, mejorando así el rendimiento del modelo.

A continuación, se muestra la expresión del descenso de gradiente:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)}), \quad (4.1)$$

donde  $\mathbf{w}^{(\tau+1)}$  representa los valores de los parámetros actualizados,  $\mathbf{w}^{(\tau)}$  representa los

<sup>1</sup>La función de pérdida es, por tanto, nuestro sistema de navegación: una puntuación baja indica que el modelo sigue el rumbo correcto, mientras que una puntuación alta sugiere que es necesario corregir la trayectoria y ajustar el rumbo del aprendizaje.

#### 4. El Dilema Clásico del Aprendizaje

valores de los parámetros antes de la actualización,  $\eta$  es un hiperparámetro, denominado **tasa de aprendizaje** o *learning rate*, que controla el tamaño del paso en cada actualización y  $\nabla E(\mathbf{w}^{(\tau)})$  indica el gradiente de la función de pérdida con respecto a los parámetros, es decir, la dirección y magnitud en la que la función de pérdida aumenta de manera más rápida.

Es importante destacar que la tasa de aprendizaje ( $\eta$ ) desempeña un papel esencial en el proceso de optimización. Si se elige un valor demasiado pequeño de la misma, el modelo podría tardar mucho en converger (véase Figura 4.2, gráfica etiquetada como “ $\eta$  too small”), requiriendo un número elevado de épocas o iteraciones para alcanzar un mínimo adecuado de la función de pérdida. Por el contrario, si se selecciona un valor demasiado grande, el modelo podría no converger e incluso divergir, oscilando alrededor del mínimo sin lograr estabilizarse o incluso aumentando la pérdida. Es por esto que, la mejor estrategia suele consistir en utilizar un valor grande al inicio del entrenamiento, para acelerar el entrenamiento, y reducirlo progresivamente a medida que avanza, con el objetivo de no divergir y estabilizarnos en torno al mínimo.

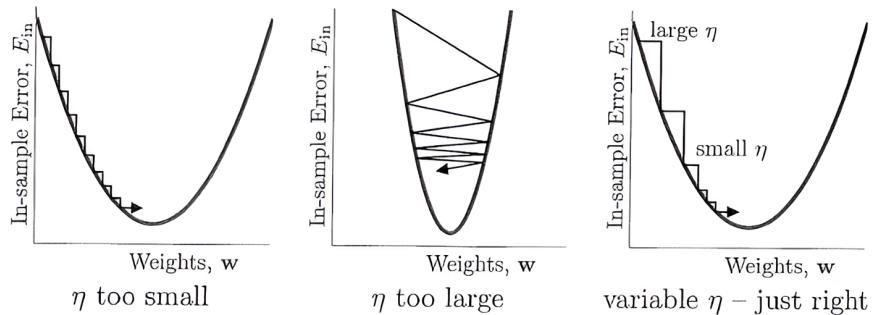


Figura 4.2.: Distintas tasas de aprendizaje para el descenso de gradiente [AMMIL12]. En la primera imagen, una tasa de aprendizaje pequeña lleva a una convergencia lenta con muchas actualizaciones. En la imagen central, una tasa demasiado grande provoca saltos bruscos que pueden impedir la convergencia. En la última imagen, una tasa variable comienza con un valor grande para avanzar rápido y disminuye progresivamente, logrando una convergencia rápida y estable.

Asimismo, existen variantes del descenso de gradiente, como el descenso de gradiente estocástico (SGD), que actualiza los parámetros utilizando cada ejemplo de entrenamiento, lo que provoca que sea muy lento cuando trabajamos con grandes volúmenes de datos. Por otro lado, nos encontramos el descenso de gradiente por lotes (Batch Gradient Descent), que utiliza el conjunto completo de datos de entrenamiento para calcular el gradiente y actualizar los parámetros, lo que permite realizar una convergencia más estable y precisa. Sin embargo, suele ser más costoso tanto computacionalmente como en términos de tiempo. Es por esto que, en el desarrollo de este proyecto, se utilizará el **descenso de gradiente por mini-lotes** (Mini-batch Gradient Descent), que combina lo mejor de ambos métodos, ya que utiliza subconjuntos de datos de entrenamiento para realizar la actualización de parámetros.

En resumen, el descenso de gradiente permite que la red neuronal mejore sus predicciones a lo largo de múltiples iteraciones o épocas. Cada vez que el modelo procesa un conjunto de datos, calcula el error y ajusta sus parámetros para aprender de los errores cometidos, repitiéndose este proceso hasta que la función de pérdida alcanza un valor mínimo aceptable.

#### 4.1.1.1. Optimizador Adam

En virtud del desarrollo de nuestro proyecto, trabajaremos con el optimizador **Adam** (*Adaptive Moment Estimation* [KB17]). Este optimizador se basa en las ideas de los algoritmos **Momentum** y **RMSProp**, combinando sus principales ventajas para lograr una convergencia más rápida y estable durante el entrenamiento de modelos de aprendizaje profundo.

El método de **Momentum** mejora al SGD acumulando un promedio móvil de los gradientes pasados. La actualización de los parámetros se realiza de la siguiente forma:

$$\begin{aligned} m_\tau &= \beta_1 \cdot m_{\tau-1} + (1 - \beta) \cdot \nabla E(w_{\tau-1}), \\ w_\tau &= w_{\tau-1} - \eta \cdot m_\tau, \end{aligned}$$

donde el término  $\beta$  actúa como un factor de decaimiento que regula la influencia de los gradientes anteriores en la actualización actual. Por su parte,  $\eta$  representa la tasa de aprendizaje. Así, la variable  $m_\tau$  corresponde al promedio móvil de los gradientes, es decir, una estimación de primer orden que suaviza la dirección de la actualización.

El algoritmo **RMSProp** adapta la tasa de aprendizaje en función de la magnitud reciente de los gradientes. Para lograrlo, calcula un promedio móvil de los cuadrados de los gradientes, lo que permite ajustar, de manera dinámica, la escala de la actualización para cada parámetro:

$$\begin{aligned} v_\tau &= \beta \cdot v_{\tau-1} + (1 - \beta)(\nabla E(w_{\tau-1})), \\ w_\tau &= w_{\tau-1} - \eta \cdot \frac{\nabla E(w_{\tau-1})}{\sqrt{v_\tau} + \epsilon}, \end{aligned}$$

donde el factor  $\beta$  actúa, al igual que en el caso anterior, como un factor de olvido que determina cuánto influyen los valores anteriores en la media móvil. Así,  $v_\tau$  representa una estimación de segundo orden, correspondiente al promedio móvil de los cuadrados de los gradientes. Además, el término  $\epsilon$  se introduce para mejorar la estabilidad numérica, evitando divisiones por cero durante la actualización de los parámetros.

Finalmente, **Adam** combina los enfoques anteriores, manteniendo simultáneamente estimaciones de primer y segundo orden. Las actualizaciones se realizan de la siguiente manera:

$$\begin{aligned} m_\tau &= \beta_1 \cdot m_{\tau-1} + (1 - \beta_1) \cdot \nabla E(w_{\tau-1}), \\ v_\tau &= \beta_2 \cdot v_{\tau-1} + (1 - \beta_2) \cdot \nabla E(w_{\tau-1}). \end{aligned}$$

Dado que  $m_\tau$  y  $v_\tau$  están inicializados a cero, se introduce una corrección de sesgo:

$$\begin{aligned} \hat{m}_\tau &= \frac{m_\tau}{1 - \beta_1^\tau}, \\ \hat{v}_\tau &= \frac{v_\tau}{1 - \beta_2^\tau}. \end{aligned}$$

Como paso final, los parámetros se actualizan de la siguiente forma:

$$w_\tau = w_{\tau-1} - \frac{\eta}{\sqrt{\hat{v}_\tau} + \epsilon} \cdot \hat{m}_\tau.$$

Este enfoque combina la dirección promedio de los gradientes con una tasa de aprendizaje adaptativa. Por un lado, acumular gradientes pasados permite suavizar la trayectoria y evitar oscilaciones, mientras que, por otro, la normalización mediante una estimación de segundo orden ajusta la tasa de aprendizaje dinámicamente para cada parámetro. Así, Adam consigue

#### 4. El Dilema Clásico del Aprendizaje

una optimización más eficiente, estable y con mejor convergencia en escenarios complejos o de alta dimensión.

##### 4.1.1.2. Aprendizaje en una red neuronal

Como se comentó en la Sección 3.2, la primera fase del aprendizaje consiste en la transmisión de la información desde la capa de entrada hacia la capa de salida, pasando por las capas ocultas, proceso conocido como propagación hacia delante o *forward pass*.

Durante este proceso, las entradas se multiplican por los pesos de la red y se suman los sesgos o *biases*, consistente en un parámetro adicional que se suma al resultado de la combinación lineal antes de pasar por la función de activación, cuya función principal es permitir que el modelo ajuste su salida de manera más flexible, sin estar forzado a pasar por el origen. Finalmente, se aplican las funciones de activación para introducir no linealidad. Este flujo de datos permite que el modelo genere una predicción para cada entrada.

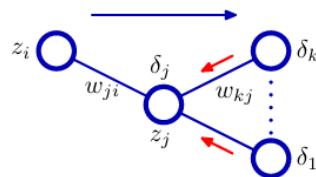


Figura 4.3.: Proceso de retropropagación del error [Biso6]. Dado un lote de entrenamiento, se propaga la información hacia delante (flecha azul), se calculan las activations y el error de salida. Seguidamente, se realiza el paso hacia atrás (flechas rojas), calculando el error  $\delta_j$  de cada unidad  $j$  en cada capa. Finalmente, se actualizan los parámetros utilizando el gradiente calculado.

Una vez obtenida la predicción, se calcula la función de pérdida para evaluar la discrepancia entre la predicción y el valor real. De esta manera, podemos describir la salida resultante de cualquier neurona mediante la siguiente expresión, extraída de [Pri23]:

$$h_d = \phi \left( w_{d0} + \sum_{i=1}^{D_i} w_{di} x_i \right),$$

donde  $d$  hace referencia a la neurona en dicha posición,  $\phi$  representa una función de activación no lineal,  $x_i \in \mathbb{R}^{D_i}$  representa la entrada multidimensional, donde  $D_i$  es el número de características de entrada (en este caso, consideramos nuestro conjunto de datos de entrenamiento como un subconjunto de  $\mathbb{R}^{D_i}$ ) y  $w_{di}$ , con  $i \in \{0, 1, \dots, D_i\}$ , representa los pesos que conectan la entrada  $x_i$  con la neurona  $d$ , donde para  $i = 0$  se obtiene el término del sesgo.

Seguidamente, se inicia la segunda fase del aprendizaje, cuyo objetivo es ajustar los parámetros de la red para minimizar ese error. Para ello, se utiliza **retropropagación del error** o *backpropagation*, que calcula el gradiente de la función de pérdida con respecto a los pesos y sesgos, mediante la regla de la cadena del cálculo diferencial. Posteriormente, el algoritmo de descenso de gradiente actualiza los pesos en la dirección opuesta al gradiente.

En conclusión, el *forward pass* y la *backpropagation* trabajan de manera conjunta para permitir que la red neuronal aprenda de manera efectiva. El *forward pass* genera las predicciones para los datos de entrada, la función de pérdida evalúa el error cometido en dichas predicciones,

la *backpropagation* calcula la contribución de cada unidad a dicho error (véase Figura 4.3) y, por último, el descenso de gradiente ajusta los parámetros para mejorar las predicciones futuras, repitiéndose este ciclo a lo largo de múltiples iteraciones.

## 4.2. Bias-variance tradeoff

El objetivo del aprendizaje radica en obtener un bajo error de test o error de generalización ( $E_{out}$ ) sobre datos desconocidos, lo que implicará que habremos conseguido una buena aproximación de la función objetivo  $f$ . La capacidad para conseguir un bajo error de generalización está ligada directamente al conjunto de hipótesis ( $\mathcal{H}$ ), donde si nuestro conjunto es suficientemente grande, tendremos una mayor probabilidad de aproximar la función objetivo, al disponer de un mayor número de funciones candidatas.

No obstante, si nuestro conjunto  $\mathcal{H}$  es demasiado grande, puede darse el caso de que, al elegir una de las funciones candidatas (usando nuestro conjunto de entrenamiento), dicha función no sea la que mejor aproxime a la función objetivo, lo que provoque un mayor error de generalización. A este problema se le conoce como el problema del **equilibrio entre aproximación y generalización**. Adicionalmente, el conjunto de hipótesis ideal sería el formado únicamente por la función objetivo, es decir,  $\mathcal{H} = \{f\}$ .

El **análisis sesgo-varianza** o *bias-variance analysis* busca descomponer el error de generalización en dos términos principales:

1. Cómo de bien puede  $\mathcal{H}$  aproximar a la función objetivo  $f$  en general, no solo en la muestra.
2. Hasta qué punto podemos acercarnos a una buena función candidata  $g \in \mathcal{H}$ .

Por otra parte, el error de generalización incluye un término adicional conocido como **ruido**. Este término hace referencia al error irreducible que se encuentra de manera natural en los datos y que, generalmente, es debido a factores fuera del control del modelo, tales como mediciones imprecisas o variables no modeladas. Dado que este tipo de error es inevitable y no puede ser reducido, no se considera relevante en la descomposición. No obstante, cabe destacar que este ruido suele ser una limitación fundamental de la generalización del modelo.

### 4.2.1. Formulación matemática del $E_{out}$

A continuación, detallaremos matemáticamente las componentes del error fuera de la muestra. Para ello y con objeto de simplificar la descomposición del  $E_{out}$  de manera limpia en los dos términos principales citados anteriormente, vamos a considerar un problema de regresión que no presenta ruido en los datos, utilizando el error cuadrático como medida de evaluación del error.

Sea  $g^{(\mathcal{D})} \in \mathcal{H}$  nuestra función candidata elegida (**hipótesis final**), que dependerá del conjunto de datos utilizado ( $\mathcal{D}$ ). Destacamos que, dado cualquier otro conjunto de datos, encontraremos una hipótesis final distinta.

El **error de generalización o error fuera de la muestra**, definido como la diferencia entre la predicción del modelo y los valores reales en un conjunto de datos no visto durante el entrenamiento, quedaría expresado de la siguiente forma:

$$E_{out}(g^{(\mathcal{D})}) = \mathbb{E}_x[(g^{(\mathcal{D})}(x) - f(x))^2], \quad x \notin \mathcal{D}, \quad (4.2)$$

#### 4. El Dilema Clásico del Aprendizaje

donde  $\mathbb{E}_x$  denota la esperanza con respecto a  $x$  (basado en la distribución de probabilidad del espacio de entrada  $\mathcal{X}$ ), con el objetivo de obtener el valor esperado del error en todo el espacio.

Dado que la ecuación (4.2) depende de un conjunto de datos en particular, podemos eliminar esta dependencia del conjunto de datos utilizando tomando el valor esperado de dicho error con respecto a todos los conjuntos de datos:

$$\mathbb{E}_{\mathcal{D}}[E_{out}(g^{(\mathcal{D})})] = \mathbb{E}_{\mathcal{D}}[\mathbb{E}_x[(g^{(\mathcal{D})}(x) - f(x))^2]].$$

Cambiamos ahora el orden de las esperanzas dado que, en realidad, estamos integrando y cambiando el orden de integración, que podemos realizarlo dado que el integrando  $(g^{(\mathcal{D})}(x) - f(x))^2$  es estrictamente no negativo:

$$\mathbb{E}_{\mathcal{D}}[E_{out}(g^{(\mathcal{D})})] = \mathbb{E}_x[\mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - f(x))^2]]. \quad (4.3)$$

Nos centramos en calcular  $\mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - f(x))^2]$ , olvidándonos por ahora de la esperanza sobre  $x$ , dado que nos interesa calcular la esperanza con respecto a  $\mathcal{D}$  hasta obtener una descomposición limpia del error. Para ello, vamos a definir el concepto de **hipótesis promedio**  $\bar{g}(x)$  de la siguiente manera:

$$\bar{g}(x) = \mathbb{E}_{\mathcal{D}}[g^{(\mathcal{D})}(x)], \quad (4.4)$$

que corresponde al valor esperado de todas las hipótesis que podemos obtener al aprender de los distintos conjuntos de datos que utilicemos. Destacamos que, en la ecuación (4.4), tenemos  $x$  (punto de prueba) fijo, por lo que  $g^{(\mathcal{D})}(x)$  es una variable aleatoria determinada por la elección de nuestros datos, donde si tomamos distintos conjuntos de datos, obtendremos distintos valores para la hipótesis en el punto  $x$  fijado.

En la realidad, nunca dispondremos de esta hipótesis promedio, pues tendríamos un número infinito de conjuntos de datos distintos. Además, cabe destacar que la hipótesis promedio no tiene asegurada su pertenencia al conjunto de hipótesis ( $\mathcal{H}$ ), aunque sea el promedio de hipótesis pertenecientes a  $\mathcal{H}$ .

Continuando con la descomposición, tenemos el siguiente resultado:

$$\mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - f(x))^2] = \mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - \bar{g}(x) + \bar{g}(x) - f(x))^2], \quad (4.5)$$

donde se ha sumado y restado la misma cantidad ( $\bar{g}(x)$ ) para simplificar la descomposición. Seguidamente, continuamos desarrollando el término cuadrático de la ecuación (4.5):

$$\mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - \bar{g}(x))^2 + (\bar{g}(x) - f(x))^2 + 2(g^{(\mathcal{D})}(x) - \bar{g}(x))(\bar{g}(x) - f(x))].$$

Dado que estamos realizando la esperanza con respecto a  $\mathcal{D}$ , de la ecuación anterior tenemos que  $(\bar{g}(x) - f(x))$  es una constante. Por tanto, dado que la esperanza de una constante es la propia constante, para obtener la esperanza del término cruzado solo necesitamos conocer la esperanza de  $(g^{(\mathcal{D})}(x) - \bar{g}(x))$ :

$$\mathbb{E}_{\mathcal{D}}[g^{(\mathcal{D})}(x) - \bar{g}(x)] = \mathbb{E}_{\mathcal{D}}[g^{(\mathcal{D})}(x)] - \mathbb{E}_{\mathcal{D}}[\bar{g}(x)] = \bar{g}(x) - \bar{g}(x) = 0.$$

Finalmente, de la ecuación (4.5) nos queda la siguiente expresión:

$$\mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - f(x))^2] = \mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - \bar{g}(x))^2] + (\bar{g}(x) - f(x))^2,$$

donde hemos usado que el valor esperado de una constante ( $\bar{g}(x) - f(x)$ ) es igual a dicha constante.

Por consiguiente, hemos obtenido una descomposición de la ecuación (4.5) en dos términos que serán los asociados a los términos de varianza (*variance*) y sesgo (*bias*), respectivamente.

1.  $\mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - \bar{g}(x))^2]$ : nos indica cómo de lejos se encuentra nuestra hipótesis,  $g^{(\mathcal{D})}(x)$ , obtenida de un conjunto de datos particular, de la mejor (promedio) hipótesis,  $\bar{g}(x)$ , que podemos obtener utilizando nuestro conjunto de hipótesis  $\mathcal{H}$ . Este es el término asociado a la varianza de  $x$  (**var(x)**).
2.  $(\bar{g}(x) - f(x))^2$ : nos indica cómo de lejos se encuentra dicha hipótesis ideal,  $\bar{g}(x)$ , de la función objetivo  $f(x)$ . Este es el término asociado al sesgo de  $x$  (**bias(x)**).

Por tanto, volviendo a la ecuación (4.3), nos queda:

$$\mathbb{E}_{\mathcal{D}}[E_{out}(g^{(\mathcal{D})})] = \mathbb{E}_x[\mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - f(x))^2]] = \mathbb{E}_x[\mathbf{var}(\mathbf{x}) + \mathbf{bias}(\mathbf{x})],$$

donde denotaremos por **sesgo** a  $\mathbb{E}_x[\mathbf{bias}(\mathbf{x})]$  y por **varianza** a  $\mathbb{E}_x[\mathbf{var}(\mathbf{x})]$ .

*Observación 4.1.* Cuando hay ruido presente en los datos de entrenamiento, es decir,  $y(x) = f(x) + \epsilon(x)$  y, además,  $\epsilon$  es una variable aleatoria con media 0 y varianza  $\sigma^2$ , entonces

$$E_{out}(g^{(\mathcal{D})}) = \mathbb{E}_{x,y}[g^{(\mathcal{D})}(x) - y(x)^2],$$

y, en este caso, se verifica:

$$\mathbb{E}_{\mathcal{D}}[E_{out}(g^{(\mathcal{D})})] = \sigma^2 + bias + var.$$

*Demostración.* Dado que asumimos que  $\epsilon$  es una variable aleatoria con media 0, obtenemos que  $\mathbb{E}[\epsilon(x)] = 0$ .

$$\mathbb{E}_{\mathcal{D},\epsilon}[(g^{(\mathcal{D})}(x) - y(x))^2] = \mathbb{E}_{\mathcal{D},\epsilon}[(g^{(\mathcal{D})}(x) - f(x)) - \epsilon(x)^2], \quad (4.6)$$

donde, dado que  $y$  depende del ruido, consideramos también el valor esperado con respecto a  $\epsilon$ , que afectará únicamente a  $y$ . Procediendo de manera similar a la ecuación (4.5), obtenemos

$$\mathbb{E}_{\mathcal{D},\epsilon}[(g^{(\mathcal{D})}(x) - \bar{g}(x) + \bar{g}(x) - f(x) - \epsilon(x))^2],$$

y, tras realizar operaciones, llegamos a

$$\mathbb{E}_{\mathcal{D},\epsilon}[(g^{(\mathcal{D})}(x) - f(x))^2] + 2\mathbb{E}_{\mathcal{D},\epsilon}[(g^{(\mathcal{D})}(x) - f(x))\epsilon(x)] + \mathbb{E}_{\mathcal{D},\epsilon}[\epsilon^2(x)],$$

donde el primer sumando no depende de  $\epsilon$ , el tercer sumando no depende de  $\mathcal{D}$  y el segundo sumando puede expresarse de la siguiente forma:

$$2\mathbb{E}_{\mathcal{D},\epsilon}[(g^{(\mathcal{D})}(x) - f(x))\epsilon(x)] = 2\mathbb{E}_{\mathcal{D},\epsilon}[(g^{(\mathcal{D})}(x) - f(x))]\mathbb{E}_{\mathcal{D},\epsilon}[\epsilon(x)],$$

donde el valor esperado con respecto a  $\epsilon$  no depende de  $\mathcal{D}$ . De esta manera y conociendo que  $\mathbb{E}_{\epsilon}[\epsilon(x)] = 0$  y  $\mathbb{E}_{\epsilon}[\epsilon(x)^2] = \sigma^2$ , obtenemos que la ecuación (4.6) puede ser reformulada como sigue:

#### 4. El Dilema Clásico del Aprendizaje

$$\mathbb{E}_{\mathcal{D}, \epsilon}[(g^{(\mathcal{D})}(x) - y(x))^2] = \mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(x) - f(x))^2] + \sigma^2.$$

Por último, aplicando la esperanza sobre  $x$  a la expresión anterior, obtenemos el resultado buscado.  $\square$

Finalmente, con el objetivo de generalizar la descomposición del error en términos de sesgo y varianza para otras funciones de pérdida, incluidas aquellas que se emplean en tareas de clasificación, recurrimos al trabajo de Pedro Domingos [Domoo].

**Definición 4.2.** Se define la **predicción óptima** ( $g(x)_*$ ) para un ejemplo  $x$  y una función de pérdida  $\mathcal{L}$  como la predicción que minimiza  $\mathbb{E}_x[\mathcal{L}(f(x), g(x)_*)]$ .

**Definición 4.3.** Definimos la **predicción principal** ( $g(x)_m$ ) para una función de pérdida  $\mathcal{L}$  sobre un conjunto de entrenamiento  $\mathcal{D}$  como

$$g(x)_m = \arg \min_{g(x)} \mathbb{E}_{\mathcal{D}}[\mathcal{L}(f(x), g(x))].$$

**Definición 4.4.** Dado un ejemplo  $x$  del conjunto de entrenamiento  $\mathcal{D}$ , se define la descomposición del error fuera de la muestra en los siguientes términos:

- **Sesgo:** El sesgo del modelo en el ejemplo  $x$  se define como

$$\mathbb{E}_x[\mathcal{L}(g(x)_*, g(x)_m)].$$

De esta forma, el sesgo es la pérdida sufrida por la predicción principal en relación con la predicción óptima, promediada entre todos los ejemplos.

- **Varianza:** La varianza en  $x$  corresponde a

$$\mathbb{E}_x[\mathbb{E}_{\mathcal{D}}[\mathcal{L}(g(x)_m, g(x))]].$$

Así, la varianza es la pérdida media sufrida por las predicciones con respecto a la predicción principal.

- **Ruido:** El ruido asociado al ejemplo  $x$  se define como

$$\mathbb{E}_x[\mathcal{L}(f(x), g(x)_*)],$$

y representa la pérdida mínima esperada debida a la incertidumbre inherente en los datos, que se produce independientemente del algoritmo de aprendizaje.

Con las definiciones anteriores, podemos descomponer el error en términos de sesgo, varianza y ruido, mediante el siguiente resultado.

**Definición 4.5.** Sea  $x \in \mathcal{D}$  un ejemplo del conjunto de entrenamiento, con valor verdadero  $f(x)$ , y sea  $g(x)$  la predicción que realiza el modelo a partir de  $\mathcal{D}$ . Para determinadas funciones de pérdida  $\mathcal{L}$  (reales, simétricas y  $\mathcal{L}(a, b) = 0 \Leftrightarrow a = b$ ), es posible expresar la descomposición del error de generalización de la siguiente forma:

$$\begin{aligned}
\mathbb{E}_{\mathcal{D},x}[\mathcal{L}(f(x), g(x))] &= c_1 \mathbb{E}_x[\mathcal{L}(f(x), g(x)_*)] \\
&\quad + \mathbb{E}_x[\mathcal{L}(g(x)_*, g(x)_m)] \\
&\quad + c_2 \mathbb{E}_x[\mathbb{E}_{\mathcal{D}}[\mathcal{L}(g(x)_m, g(x))]] \\
&= c_1 \mathbf{noise}(x) + \mathbf{sesgo}(x) + c_2 \mathbf{var}(x),
\end{aligned} \tag{4.7}$$

donde  $c_1$  y  $c_2$  son constantes multiplicativas que tomarán distintos valores para las diferentes funciones de pérdida.

*Observación 4.6.* La ecuación (4.7) se cumple para la función de pérdida cuadrática, con  $c_1 = c_2 = 1$ , como se ha demostrado anteriormente.

**Teorema 4.7.** *La ecuación (4.7) es válida para la pérdida cero-uno ( $\mathcal{L}_{0-1}(a, b) = \chi_{[a \neq b]}$ ) en problemas de clasificación con dos clases, con  $c_1 = 2P_{\mathcal{D}}[g(x) = g(x)_*] - 1$  y  $c_2 = 1$  si  $g(x)_m = g(x)_*$ ,  $c_2 = -1$  en otro caso.*

*Demostración.* Comenzamos la demostración mostrando que

$$\mathbb{E}_x[\mathcal{L}(f(x), g(x))] = \mathcal{L}(g(x)_*, g(x)) + c_0 \mathbb{E}_x[\mathcal{L}(f(x), g(x)_*)], \tag{4.8}$$

con  $c_0 = 1$  si  $g(x) = g(x)_*$  y  $c_0 = -1$  si  $g(x) \neq g(x)_*$ , y donde  $\mathcal{L}(a, b)$  representa la pérdida cero-uno. Si  $g(x) = g(x)_*$ , la ecuación (4.8) es cierta de manera trivial usando  $c_0 = 1$ . Supongamos ahora que  $g(x) \neq g(x)_*$ . Dado que solo hay dos clases, si  $g(x) \neq g(x)_*$  entonces  $f(x) \neq g(x)$  implica que  $f(x) = g(x)_*$  y viceversa. Por tanto,  $P_x[f(x) = g(x)] = P_x[f(x) = g(x)_*]$ , y se tiene

$$\begin{aligned}
\mathbb{E}_x[\mathcal{L}(f(x), g(x))] &= P_x[f(x) \neq g(x)] = 1 - P_x[f(x) = g(x)] \\
&= 1 - P_x[f(x) = g(x)_*] = 1 - \mathbb{E}_x[\mathcal{L}(f(x), g(x)_*)] \\
&= \mathcal{L}(g(x)_*, g(x)) - \mathbb{E}_x[\mathcal{L}(f(x), g(x)_*)] \\
&= \mathcal{L}(g(x)_*, g(x)) + c_0 \mathbb{E}_x[\mathcal{L}(f(x), g(x)_*)],
\end{aligned}$$

con  $c_0 = -1$ , demostrando así la ecuación (4.8). De manera similar, mostramos que

$$\mathbb{E}_{\mathcal{D}}[\mathcal{L}(g(x)_*, g(x))] = \mathcal{L}(g(x)_*, g(x)_m) + c_2 \mathbb{E}_{\mathcal{D}}[\mathcal{L}(g(x)_m, g(x))], \tag{4.9}$$

con  $c_2 = 1$  si  $g(x)_m = g(x)_*$  y  $c_2 = -1$  si  $g(x)_m \neq g(x)_*$ . Si  $g(x)_m = g(x)_*$ , la ecuación (4.9) es trivialmente cierta con  $c_2 = 1$ . Si  $g(x)_m \neq g(x)_*$ , entonces  $g(x)_m \neq g(x)$  implica que  $g(x)_* = g(x)$  y viceversa, y se cumple

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}[\mathcal{L}(g(x)_*, g(x))] &= P_{\mathcal{D}}[g(x)_* \neq g(x)] = 1 - P_{\mathcal{D}}[g(x)_* = g(x)] \\
&= 1 - P_{\mathcal{D}}[g(x)_m = g(x)] = 1 - \mathbb{E}_{\mathcal{D}}[\mathcal{L}(g(x)_m, g(x))] \\
&= \mathcal{L}(g(x)_*, g(x)_m) - \mathbb{E}_{\mathcal{D}}[\mathcal{L}(g(x)_m, g(x))] \\
&= \mathcal{L}(g(x)_*, g(x)_m) + c_2 \mathbb{E}_{\mathcal{D}}[\mathcal{L}(g(x)_m, g(x))],
\end{aligned}$$

con  $c_2 = -1$ , lo que demuestra la ecuación (4.9). A continuación, usando la ecuación (4.8), obtenemos:

#### 4. El Dilema Clásico del Aprendizaje

$$\mathbb{E}_{\mathcal{D},x}[\mathcal{L}(f(x), g(x))] = \mathbb{E}_{\mathcal{D}}[\mathbb{E}_x[\mathcal{L}(f(x), g(x))]] = \mathbb{E}_{\mathcal{D}}[\mathcal{L}(g(x)_*, g(x)) + c_0 \mathbb{E}_x[\mathcal{L}(f(x), g(x)_*)]].$$

Dado que  $L(f(x), g(x)_*)$  no depende de  $D$ , se verifica:

$$\mathbb{E}_{\mathcal{D},x}[\mathcal{L}(f(x), g(x))] = \mathbb{E}_{\mathcal{D}}[\mathcal{L}(g(x)_*, g(x))] + \mathbb{E}_{\mathcal{D}}[c_0] \mathbb{E}_x[\mathcal{L}(f(x), g(x)_*)],$$

y dado que

$$\mathbb{E}_{\mathcal{D}}[c_0] = P_{\mathcal{D}}[g(x) = g(x)_*] - P_{\mathcal{D}}[g(x) \neq g(x)_*] = 2P_{\mathcal{D}}[g(x) = g(x)_*] - 1 = c_1,$$

se obtiene finalmente la ecuación (4.7), usando la ecuación (4.9).  $\square$

No obstante, cabe resaltar que existen funciones de pérdida para las cuales la ecuación (4.7) no se cumple de manera exacta. Sin embargo, siempre que la función de pérdida sea una métrica, es posible acotar el error esperado superior e inferiormente mediante los términos de ruido, sesgo y varianza, como se muestra en el siguiente resultado.

**Teorema 4.8.** *Las siguientes desigualdades son válidas para cualquier función de pérdida  $\mathcal{L}$  que sea a su vez una métrica:*

$$\begin{aligned}\mathbb{E}_{\mathcal{D},x}[\mathcal{L}(f(x), g(x))] &\leq \mathbf{noise}(x) + \mathbf{sesgo}(x) + \mathbf{var}(x) \\ \mathbb{E}_{\mathcal{D},x}[\mathcal{L}(f(x), g(x))] &\geq \max \left\{ \mathbf{noise}(x) - \mathbf{sesgo}(x) - \mathbf{var}(x), \right. \\ &\quad \mathbf{sesgo}(x) - \mathbf{var}(x) - \mathbf{noise}(x), \\ &\quad \left. \mathbf{var}(x) - \mathbf{sesgo}(x) - \mathbf{noise}(x) \right\}.\end{aligned}$$

*Demostración.* Utilizando la desigualdad triangular propia de una métrica, obtenemos

$$\begin{aligned}\mathcal{L}(f(x), g(x)) &\leq \mathcal{L}(f(x), g(x)_*) + \mathcal{L}(g(x)_*, g(x)) \\ &\leq \mathcal{L}(f(x), g(x)_*) + \mathcal{L}(g(x)_*, g(x)_m) + \mathcal{L}(g(x)_m, g(x)).\end{aligned}$$

Tomando el valor esperado de esta ecuación respecto de  $\mathcal{D}$  y  $x$ , y simplificando, se obtiene la cota superior. Usando nuevamente la desigualdad triangular y la simetría:

$$\begin{aligned}\mathcal{L}(g(x)_*, g(x)_m) &\leq \mathcal{L}(g(x)_*, f(x)) + \mathcal{L}(f(x), g(x)) + \mathcal{L}(g(x), g(x)_m) \\ &\leq \mathcal{L}(f(x), g(x)_*) + \mathcal{L}(f(x), g(x)) + \mathcal{L}(g(x)_m, g(x)).\end{aligned}$$

Reordenando términos, tomando la esperanza respecto de  $\mathcal{D}$  y  $x$ , y simplificando:

$$\mathbb{E}_{\mathcal{D},x}[\mathcal{L}(f(x), g(x))] \geq \mathbf{sesgo}(x) - \mathbf{var}(x) - \mathbf{noise}(x).$$

Los términos restantes que aparecen en la cota inferior pueden derivarse de forma análoga.  $\square$

Como conclusión de esta sección, detallaremos algunos aspectos clave del análisis del sesgo y de la varianza:

- Aunque el análisis de sesgo-varianza se basa principalmente en la medida del error cuadrático, el algoritmo de aprendizaje utilizado por el modelo no tiene que basarse en minimizar el error cuadrático. Es decir, se puede usar cualquier otro criterio (función de pérdida) para producir  $g^{(\mathcal{D})}$  basado en el conjunto de datos  $\mathcal{D}$  y, una vez que tenemos  $g^{(\mathcal{D})}$ , calculamos su sesgo y varianza.
- El sesgo y la varianza **no pueden ser calculados en la práctica**, ya que dependen de la función objetivo y de la distribución de probabilidad de la entrada (ambas desconocidas), por lo que su descomposición resulta de utilidad como una herramienta conceptual a la hora de entender y desarrollar el comportamiento del error.

### 4.3. Equilibrio clásico entre sesgo y varianza

Siguiendo con los resultados obtenidos en la Subsección 4.2.1, nuestro objetivo ahora es minimizar el error fuera de la muestra, inducido por tres componentes: ruido, sesgo y varianza. Dado que, como se ha comentado previamente, el ruido es irreducible y no hay nada que podamos hacer para evitarlo, nos centraremos en intentar reducir los dos términos principales reducibles del error: el sesgo y la varianza.

Conocemos que el sesgo viene definido por la medida en la que la predicción ideal obtenida de todos los conjuntos de datos difiere de la función objetivo, o expresado de manera similar, el sesgo es el resultado de la incapacidad del modelo para describir la función objetivo. Este resultado sugiere que podemos reducir este término de error haciendo que nuestro modelo sea más flexible, es decir, aumentando nuestro conjunto de hipótesis  $\mathcal{H}$  (haciéndolo más complejo), con el objetivo de disponer de un mayor número de funciones candidatas para forzar que la hipótesis promedio se aproxime lo máximo posible a la función objetivo.

Por otro lado, la varianza viene a ofrecernos la medida en la que varían las hipótesis para distintos conjuntos de datos con respecto a la hipótesis ideal, o expresado de manera similar, la varianza evalúa cómo de sensible es una hipótesis a la selección específica del conjunto de datos de entrenamiento  $\mathcal{D}$ . Este análisis revela que podemos reducir este término de error disminuyendo el conjunto de hipótesis ya que, en un espacio de hipótesis restringido, al haber menos hipótesis, estas son menos sensibles a las variaciones en el conjunto de datos. Como resultado, al cambiar el conjunto de datos para seleccionar una de ellas, es más probable que se elijan hipótesis similares.

En consecuencia, observamos una dependencia de ambos términos de error con respecto a la **complejidad del conjunto de hipótesis ( $\mathcal{H}$ )** utilizado. Esta dependencia viene dada por:

- Al aumentar la complejidad del conjunto de hipótesis, es decir, al incrementar su tamaño, el sesgo disminuirá, pero la varianza irá aumentado.
- Si reducimos la complejidad del conjunto de hipótesis, es decir, disminuimos su tamaño, el sesgo irá aumentando, pero la varianza disminuirá.

Esta tensión inherente entre complejidad y simplicidad es conocida como el *bias-variance tradeoff*, puesto que aumentar la complejidad del modelo tiende a reducir el sesgo, pero incrementa la varianza, y viceversa (véase Figura 4.4). Por tanto, alcanzar un balance adecuado entre ambos es clave para conseguir una buena capacidad de generalización.

#### 4. El Dilema Clásico del Aprendizaje

Como conclusión, el equilibrio buscado en esta sección, con el objetivo de minimizar el error fuera de la muestra, está ligado a la elección de un conjunto de hipótesis ( $\mathcal{H}$ ) que sea lo suficientemente complejo como para aproximarse a la función objetivo, pero también lo suficientemente simple como para evitar una cantidad excesiva de hipótesis que induzcan un alto grado de varianza. Este equilibrio se puede conseguir mediante diversas técnicas de regularización, que buscan encontrar este punto óptimo de complejidad adecuado para nuestro problema.

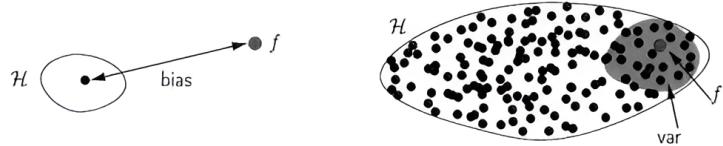


Figura 4.4.: Distintos casos del conjunto de hipótesis y de la función objetivo [AMMIL12]. A la izquierda, observamos como nuestro conjunto de hipótesis presenta únicamente una función candidata, alejada de la función objetivo  $f$ , lo que implica un alto sesgo y una varianza nula. A la derecha, vemos un conjunto de hipótesis con numerosas funciones candidatas que incluye a la función objetivo, por lo que el sesgo es muy cercano a cero y la varianza es grande.

##### 4.3.1. Curva de aprendizaje

Para finalizar esta sección, introduciremos el concepto de curva de aprendizaje o *learning curve*, que será la principal herramienta que utilizaremos a lo largo de todo el proyecto para analizar el rendimiento de los distintos modelos que utilicemos.

En primer lugar y de manera análoga a la ecuación (4.2), definimos el **error de entrenamiento o error dentro de la muestra** ( $E_{in}$ ) como la diferencia entre la predicción del modelo y los valores reales del conjunto de datos de entrenamiento, es decir:

$$E_{in}(g^{(\mathcal{D})}) = \mathbb{E}_x[(g^{(\mathcal{D})}(x) - f(x))^2], \quad x \in \mathcal{D}.$$

Por consiguiente, después de aprender de un conjunto particular de datos  $\mathcal{D}$  de tamaño  $N$ , la hipótesis final elegida  $g^{(\mathcal{D})}$  tendrá error de entrenamiento ( $E_{in}(g^{(\mathcal{D})})$ ) y error de generalización ( $E_{out}(g^{(\mathcal{D})})$ ), ambos dependiendo del conjunto de datos utilizado. Como se comentó en la Subsección 4.2.1, al realizar la esperanza con respecto a todos los conjuntos de datos de dichos errores, obtenemos los errores esperados  $\mathbb{E}_{\mathcal{D}}[E_{in}(g^{(\mathcal{D})})]$  y  $\mathbb{E}_{\mathcal{D}}[E_{out}(g^{(\mathcal{D})})]$ , los cuales son funciones del tamaño del conjunto de datos ( $N$ ).

Se denomina **curva de aprendizaje** a la representación gráfica que muestra la relación entre el rendimiento de un modelo y el tamaño del conjunto de datos utilizado para su entrenamiento, es decir, a la gráfica que incluye los errores esperados  $\mathbb{E}_{\mathcal{D}}[E_{in}(g^{(\mathcal{D})})]$  y  $\mathbb{E}_{\mathcal{D}}[E_{out}(g^{(\mathcal{D})})]$  como función de  $N$ .

Además, dado que la curva de aprendizaje está ligada a los errores esperados del modelo tanto dentro como fuera de la muestra, podríamos analizar el equilibrio sesgo-varianza directamente sobre dicha gráfica. Sin embargo, para llevar a cabo dicho análisis, necesitaríamos conocer la hipótesis promedio  $\bar{g}$  que, como sabemos de secciones anteriores, es imposible de calcular. No obstante, si tuvieramos dicha hipótesis promedio, podríamos realizar el análisis

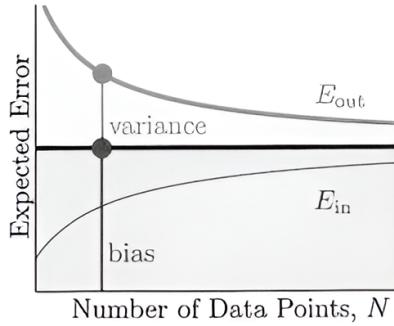


Figura 4.5.: Ejemplo de curva de aprendizaje tradicional [AMMIL12]. Observamos la curva de aprendizaje de un modelo con respecto al tamaño del conjunto de datos. Se puede comprobar como el  $E_{out}$  decrece, mientras que el  $E_{in}$  crece en función de  $N$ . Además, se puede apreciar la descomposición sesgo-varianza, donde la línea central en negrita denota la hipótesis promedio.

(véase Figura 4.5), teniendo en cuenta que el  $E_{out}$  es suma de sesgo y varianza (obviando el término de ruido).

Sin embargo, de cara a analizar el doble descenso a lo largo del proyecto, **no utilizaremos directamente la curva de aprendizaje** tal y como se ha definido, dado que nuestro objetivo es analizar el error con respecto a la capacidad del modelo y no en función del número de datos utilizados. Es por esto que, nuestra curva de aprendizaje será una modificación de la curva de aprendizaje original, donde en el eje X de la gráfica aparecerá, indistintamente, la capacidad del modelo, y en el eje Y el error esperado (véase Figura 4.6), para un número fijo de datos de entrenamiento.

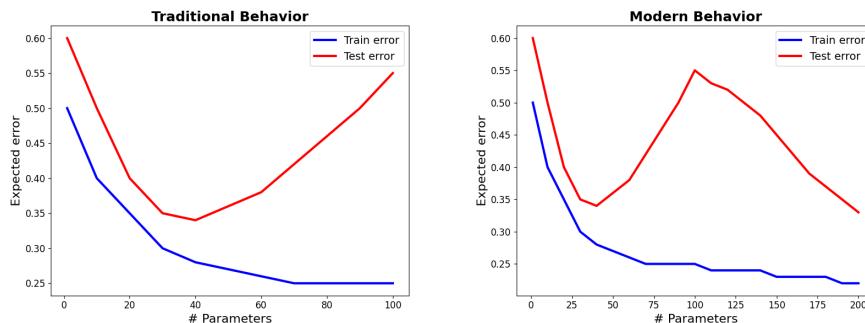


Figura 4.6.: Ejemplos de curvas de aprendizaje modificadas para este proyecto. A la izquierda, se muestra la curva clásica de aprendizaje, donde el error de test aumenta, de manera progresiva, una vez superado un cierto punto. A la derecha, la curva moderna refleja que, al aumentar la complejidad del modelo más allá de lo establecido en la sabiduría clásica, el error de test vuelve a disminuir.

#### 4.4. Underfitting y overfitting

Consideremos nuevamente el problema del aprendizaje supervisado que estamos tratando, consistente en encontrar una “buena” función aproximadora  $g \in \mathcal{H}$  basada en los datos de un determinado conjunto de entrenamiento  $\mathcal{D}$ . Además, se asume que estos datos provienen de una determinada distribución de probabilidad, es decir,  $\mathcal{D} = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$  es una colección de  $n$  copias idénticas e idénticamente distribuidas de las variables aleatorias  $(X, Y)$  que toman valores en  $\mathcal{X} \times \mathcal{Y}$ , y que siguen una distribución de probabilidad conjunta  $P[X, Y]$ , permitiendo modelar la incertidumbre en las predicciones. De igual manera, se asume la existencia de una función real no negativa  $\mathcal{L}(g(X), Y)$ , denominada **función de pérdida**, que es la encargada de evaluar la diferencia entre la predicción realizada por la función candidata  $g$  y el verdadero valor  $y$ .

En lo que prosigue a lo largo de esta sección, restringiremos nuestro conjunto de hipótesis  $\mathcal{H}$  a una determinada clase de funciones, es decir,  $\mathcal{H}$  quedaría fijo y trabajaremos en el contexto de la sabiduría tradicional, esto es, cuando el número de parámetros del modelo es significativamente menor que el número de datos de entrenamiento disponibles.

**Definición 4.9** (Riesgo real). El riesgo real asociado a la función candidata  $g \in \mathcal{H}$  viene definido como el valor esperado de la función de pérdida, esto es

$$\mathcal{L}(g) = \mathbb{E}[\mathcal{L}(g(X), Y)] = \int_{\mathcal{X}} \mathcal{L}(g(X), Y) dP[X, Y] = P[g(X) \neq Y].$$

El riesgo real es también denominado como error de generalización.

Por tanto, el objetivo final de un algoritmo de aprendizaje, como se comentó en secciones anteriores, es encontrar la función candidata  $g^*$  entre una clase fija de funciones del conjunto de hipótesis  $\mathcal{H}$  para la cual el riesgo real sea mínimo:

$$g^* = \arg \min_{g \in \mathcal{H}} \mathcal{L}(g).$$

No obstante, en la práctica y por lo general, el riesgo real no puede ser calculado porque la distribución conjunta  $P[X, Y]$  es desconocida por el algoritmo de aprendizaje. Es por esto que tenemos que recurrir a un cálculo estimado del mismo, denominado **riesgo empírico**, calculado haciendo uso de la media de la función de pérdida sobre el conjunto de entrenamiento.

**Definición 4.10** (Riesgo empírico). El riesgo empírico asociado a la función candidata  $g \in \mathcal{H}$  sobre el conjunto de entrenamiento  $\mathcal{D}$  viene definido por:

$$\mathcal{L}_{emp}(g) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(g(X_i), Y_i).$$

Ligado a este concepto surge el **principio de minimización del riesgo empírico** [Vap91], que establece que el algoritmo de aprendizaje  $\mathcal{A}$  debe elegir una función candidata  $\hat{g}$  sobre el conjunto de hipótesis  $\mathcal{H}$  que minimice el riesgo empírico sobre el conjunto de entrenamiento  $\mathcal{D}$ , es decir

$$\hat{g} = \arg \min_{g \in \mathcal{H}} \mathcal{L}_{emp}(g). \tag{4.10}$$

Finalmente, la diferencia entre cualquier función candidata  $g \in \mathcal{H}$  y la mejor función candidata  $g^*$  puede descomponerse de la siguiente manera [LT24]:

$$\mathcal{L}(g) - \mathcal{L}(g^*) = \underbrace{\mathcal{L}(g) - \inf_{g \in \mathcal{H}} \mathcal{L}(g)}_{\text{error de estimación}} + \underbrace{\inf_{g \in \mathcal{H}} \mathcal{L}(g) - \mathcal{L}(g^*)}_{\text{error de aproximación}},$$

donde, aparte del error de estimación y del error de aproximación, existe otra fuente de error conocida como **error de optimización**, que indica la diferencia entre el riesgo de la función candidata, devuelto por el procedimiento de optimización (en nuestro caso el descenso de gradiente), y un minimizador del riesgo empírico. De este modo, el algoritmo de aprendizaje  $\mathcal{A}$ , basado en el principio de minimización del riesgo empírico, se define como la solución del problema de optimización planteado en la ecuación (4.10).

**Proposición 4.11.** *Para cualquier minimizador del riesgo empírico  $\hat{g}$ , el error de estimación verifica*

$$\mathcal{L}(\hat{g}) - \inf_{g \in \mathcal{H}} \mathcal{L}(g) \leq 2 \sup_{g \in \mathcal{H}} |\mathcal{L}_{emp}(g) - \mathcal{L}(g)|.$$

*Demostración.* Partimos de la siguiente desigualdad

$$\mathcal{L}(\hat{g}) - \inf_{g \in \mathcal{H}} \mathcal{L}(g) \leq |\mathcal{L}(\hat{g}) - \mathcal{L}_{emp}(\hat{g})| + |\mathcal{L}_{emp}(\hat{g}) - \inf_{g \in \mathcal{H}} \mathcal{L}(g)|,$$

con el primer sumando verificando

$$|\mathcal{L}(\hat{g}) - \mathcal{L}_{emp}(\hat{g})| \leq \sup_{g \in \mathcal{H}} |\mathcal{L}_{emp}(g) - \mathcal{L}(g)|,$$

dado que  $\hat{g} \in \mathcal{H}$ . Por otra parte, el segundo sumando verifica

$$|\mathcal{L}_{emp}(\hat{g}) - \inf_{g \in \mathcal{H}} \mathcal{L}(g)| = |\inf_{g \in \mathcal{H}} \mathcal{L}_{emp}(g) - \inf_{g \in \mathcal{H}} \mathcal{L}(g)| \leq \sup_{g \in \mathcal{H}} |\mathcal{L}_{emp}(g) - \mathcal{L}(g)|,$$

y, sumando ambas desigualdades, se obtiene el resultado deseado.  $\square$

De este modo, la estrategia clásica a seguir en el aprendizaje automático es la de encontrar un correcto conjunto de hipótesis  $\mathcal{H}$  para mantener ambos errores lo más pequeños posible. Es por esto que, dependiendo del conjunto  $\mathcal{H}$  elegido, podemos encontrarnos las siguientes situaciones:

1. Si el conjunto  $\mathcal{H}$  es muy “pequeño”, ninguna función candidata será capaz de capturar la complejidad de los datos de entrenamiento y no será capaz de aproximarse a  $g^*$ . A esta situación la llamaremos **subajuste** o *underfitting*.
2. Si el conjunto  $\mathcal{H}$  es muy “grande”, el límite de la Proposición 4.11 (máxima brecha de generalización sobre  $\mathcal{H}$ ) aumentará, y la función candidata  $\hat{g}$  elegida como minimizadora del riesgo empírico puede generalizar de forma no adecuada aún teniendo un error de entrenamiento bajo. A esta situación la llamaremos **sobreajuste** o *overfitting*.

Como conclusión de estas situaciones, se pone de manifiesto la fuerte dependencia de la minimización empírica del riesgo (error del modelo) del conjunto de hipótesis elegido, basado en la sabiduría convencional. Además, es posible establecer de manera precisa la

#### 4. El Dilema Clásico del Aprendizaje

relación entre la descomposición del error de generalización como suma de sesgo y varianza y los fenómenos de *underfitting* y *overfitting* (véase Figura 4.7). De esta manera, cuando el modelo se encuentra en la zona de subajuste, su estructura es demasiado simple y no logra capturar los patrones presentes en los datos. Esto se traduce en un error elevado tanto en el conjunto de entrenamiento como en el de test, ligado a un alto sesgo.

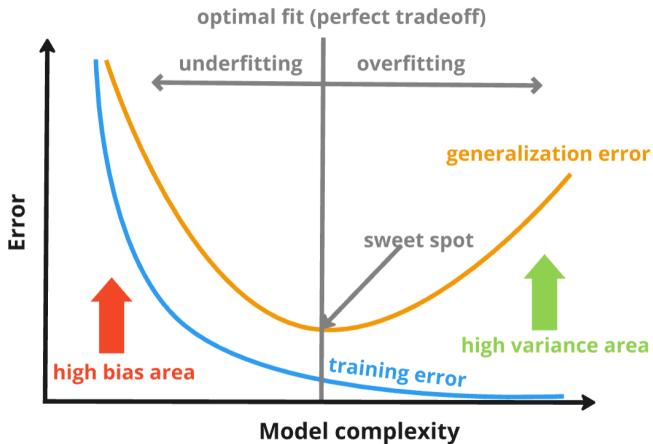


Figura 4.7.: Relación *bias/variance* con *underfitting* y *overfitting* en el contexto clásico. Al principio, el modelo muestra un alto sesgo, incapaz de capturar adecuadamente las relaciones entre los datos. A medida que aumenta su complejidad, su rendimiento mejora hasta alcanzar un punto de equilibrio en el mínimo de la curva del error de generalización (*sweet spot*). Al superar este punto óptimo, el modelo comienza a memorizar los datos, lo que provoca un aumento del error de generalización, ligado a una mayor varianza.

Por otro lado, cuando el modelo se encuentra en la zona de sobreajuste, se ajusta en exceso a los datos de entrenamiento, llegando a memorizar el ruido. En este caso, el modelo presenta un desempeño prácticamente perfecto en el conjunto de entrenamiento, pero su capacidad de generalización a nuevos datos es deficiente, ligada a una varianza elevada y un bajo nivel de sesgo.

En definitiva, para minimizar el error de generalización en la zona clásica, es fundamental encontrar un equilibrio entre sesgo y varianza, evitando tanto el *underfitting* como el *overfitting*. Este equilibrio se logra mediante una elección adecuada del conjunto de hipótesis  $\mathcal{H}$ , el cual debe ser lo suficientemente expresivo para capturar patrones relevantes en los datos sin llegar a ser demasiado complejo para que termine modelando el ruido presente en los mismos.

## 4.5. Marcos de generalización clásicos

Tras haber presentado la descomposición del error fuera de la muestra en términos de sesgo y varianza, en esta sección introducimos un enfoque complementario para analizar el equilibrio entre aproximación y generalización.

Comenzamos presentando un resultado fundamental que nos permite caracterizar la diferencia entre el error fuera de la muestra y el error dentro de la muestra mediante una cota superior probabilística.

**Definición 4.12** (Desigualdad de Hoeffding). Sean  $n$  el número de ejemplos en el conjunto de entrenamiento  $\mathcal{D}$ ,  $h \in \mathcal{H}$  una hipótesis fija y  $\epsilon > 0$ . Entonces, se cumple:

$$P [|E_{in}(h) - E_{out}(h)| > \epsilon] \leq 2e^{-2\epsilon^2 n}.$$

Esta desigualdad nos indica que, a medida que el tamaño de la muestra aumenta, la probabilidad de que el error dentro de la muestra ( $E_{in}$ , que es una variable aleatoria que depende de la muestra) se desvíe del error fuera de la muestra ( $E_{out}$ , que es una constante desconocida pero no aleatoria) en más de una cantidad  $\epsilon$ , decrece de forma exponencial. Es decir, es cada vez menos probable que exista una gran discrepancia entre el rendimiento observado en el entrenamiento y el rendimiento real.

No obstante, nuestro interés radica en obtener una cota para la hipótesis final que elige el algoritmo de aprendizaje, y no únicamente para una hipótesis fija. Por esta razón, consideramos un conjunto finito de hipótesis  $\mathcal{H}$  en lugar de una sola, de modo que:

$$\mathcal{H} = \{h_1, h_2, \dots, h_M\},$$

donde  $M$  representa la cardinalidad del conjunto. En este contexto, si queremos acotar la probabilidad de que se produzca una desviación entre el error dentro y fuera de la muestra para la hipótesis seleccionada  $g$ , dicha desviación debe ocurrir para al menos una de las hipótesis del conjunto. Por tanto:

$$|E_{in}(g) - E_{out}(g)| > \epsilon \Rightarrow \left( \begin{array}{l} |E_{in}(h_1) - E_{out}(h_1)| > \epsilon \\ \textbf{or} \quad |E_{in}(h_2) - E_{out}(h_2)| > \epsilon \\ \dots \\ \textbf{or} \quad |E_{in}(h_M) - E_{out}(h_M)| > \epsilon \end{array} \right).$$

Aplicando el hecho de que si  $\mathcal{B}_1 \Rightarrow \mathcal{B}_2$  entonces  $P[\mathcal{B}_1] \leq P[\mathcal{B}_2]$ , y utilizando además que la probabilidad de la unión de eventos está acotada por la suma de sus probabilidades individuales, obtenemos:

$$P [|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 n}, \quad (4.11)$$

donde  $g$  es una hipótesis elegida por el algoritmo de aprendizaje dentro del conjunto  $\mathcal{H}$ . Cabe señalar que esta cota es más laxa que la que se obtiene para una hipótesis fija, ya que considera el peor caso sobre todo el conjunto  $\mathcal{H}$ . Sin embargo, dicha acotación sigue siendo útil siempre que  $M$  sea finito, puesto que, en ese caso, el crecimiento de la cota está controlado y permite garantizar una buena generalización con alta probabilidad.

Por otra parte, la caracterización dada en la ecuación (4.11) puede reformularse en términos de una cota probabilística más explícita. Para un valor de tolerancia  $\delta \in (0, 1)$ , se puede afirmar que, con probabilidad al menos  $1 - \delta$ , se verifica:

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2n} \log \left( \frac{2M}{\delta} \right)},$$

donde se proporciona una cota superior para el error fuera de la muestra en función del error dentro de la muestra, el tamaño del conjunto de entrenamiento  $n$ , la cardinalidad del conjunto de hipótesis  $M$  y el nivel de confianza  $1 - \delta$ .

A continuación, extendemos la desigualdad de Hoeffding para el caso en que  $\mathcal{H}$  es infinito. Para ello, buscamos reemplazar el valor de  $M$  por uno que sea finito.

#### 4. El Dilema Clásico del Aprendizaje

**Definición 4.13.** Sean  $x_1, \dots, x_n \in \mathcal{X}$ . Las dicotomías generadas por  $\mathcal{H}$  en estos puntos vienen dadas por:

$$\mathcal{H}(x_1, \dots, x_n) = \{h(x_1), \dots, h(x_n) \mid h \in \mathcal{H}\}.$$

De esta forma, las dicotomías  $\mathcal{H}(x_1, \dots, x_n)$  forman un conjunto de hipótesis, con la salvedad de que estas hipótesis solo se observan a través de los  $n$  ejemplos de entrenamiento. Un mayor conjunto  $\mathcal{H}(x_1, \dots, x_n)$  implica que  $\mathcal{H}$  es más diverso y genera más dicotomías en dichos puntos.

**Definición 4.14.** La función de crecimiento para un conjunto de hipótesis  $\mathcal{H}$  se define como:

$$m_{\mathcal{H}}(n) = \max_{x_1, \dots, x_n \in \mathcal{X}} |\mathcal{H}(x_1, \dots, x_n)|,$$

donde  $|\cdot|$  denota el cardinal del conjunto.

Así,  $m_{\mathcal{H}}(n)$  es el máximo número de dicotomías que pueden ser generadas por  $\mathcal{H}$  en cualesquiera  $n$  puntos. Al igual que el valor  $M$  en la desigualdad de Hoeffding,  $m_{\mathcal{H}}(n)$  es una medida del número de hipótesis en  $\mathcal{H}$ , con la salvedad de que no se tiene en cuenta todo el espacio muestral  $\mathcal{X}$ .

**Definición 4.15** (Dimensión VC). La **dimensión de Vapnik-Chervonenkis (VC)** de un conjunto de hipótesis  $\mathcal{H}$  sobre funciones objetivo binarias (donde cada  $h \in \mathcal{H}$  asigna valores en  $\{-1, 1\}$ ), denotada por  $d_{vc}(\mathcal{H})$ , es el valor más grande de  $n$  tal que  $m_{\mathcal{H}}(n) = 2^n$ . Además, si  $m_{\mathcal{H}}(n) = 2^n$  para todo  $n$ , entonces  $d_{vc}(\mathcal{H}) = \infty$ .

Aunque esta definición se basa en funciones binarias, puede extenderse a otros tipos de funciones. De esta forma, esta dimensión mide la complejidad de un conjunto de hipótesis en términos de su capacidad para ajustarse o “destrozar” puntos en diferentes configuraciones. En otras palabras, la dimensión VC nos indica cuántos puntos pueden ser completamente separados en todas las posibles combinaciones por las funciones del conjunto de hipótesis.

Finalmente, reemplazando el valor de  $M$  por  $m_{\mathcal{H}}(n)$  en la desigualdad de Hoeffding, obtenemos el siguiente límite de generalización, que es válido para cualquier función objetivo binaria  $f$ , cualquier conjunto de hipótesis  $\mathcal{H}$ , cualquier algoritmo de aprendizaje, y cualquier distribución de probabilidad de entrada.

**Teorema 4.16** (Límite de generalización VC). *Para cualquier tolerancia  $\delta > 0$ , se verifica:*

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{n} \ln \left( \frac{4m_{\mathcal{H}}(2n)}{\delta} \right)},$$

con probabilidad al menos  $1 - \delta$ .

Este resultado implica que si la dimensión VC es finita, el término en la parte derecha de la desigualdad converge hacia cero, ya que  $m_{\mathcal{H}}(2n)$  es polinomial de orden  $d_{vc}$  en  $n$ . En consecuencia, cualquier hipótesis de un conjunto de hipótesis infinito con una dimensión VC finita generalizará bien, es decir,  $E_{out}(g)$  será cercano a  $E_{in}(g)$ .

En la mayoría de los casos prácticos, contamos con un conjunto de entrenamiento fijo, por lo que el tamaño de la muestra  $n$  es constante. En este contexto, nuestra principal preocupación es determinar el rendimiento real que podemos esperar sobre este conjunto de datos. Para abordar esta cuestión, el Teorema 4.16 se puede expresar en términos de la dimensión VC ( $d_{vc}$ ) en lugar de  $m_{\mathcal{H}}(2n)$ , lo que da lugar a una cota más comúnmente utilizada en la práctica:

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{n} \ln \left( \frac{4(2n)^{d_{vc}} + 1}{\delta} \right)},$$

donde el error fuera de la muestra queda acotado por el error dentro de la muestra y un término adicional que depende de la dimensión VC de  $\mathcal{H}$ . Este término crece a medida que la dimensión VC de  $\mathcal{H}$  aumenta, penalizando la acotación del error de generalización debido al impacto de la complejidad del modelo (véase Figura 4.8). De esta manera, el modelo óptimo representa un equilibrio que minimiza la combinación de ambos términos.

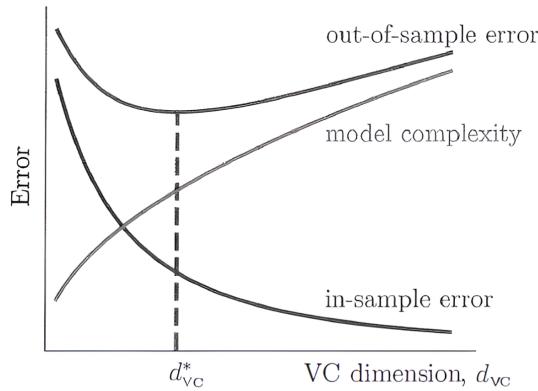


Figura 4.8.: Error en función de la dimensión VC [AMMIL12]. Al utilizar un modelo más complejo (con mayor  $d_{vc}$ ) nos adaptamos mejor a los datos de entrenamiento, pero a cambio sufrimos una penalización mayor en la generalización por la complejidad del modelo. La combinación de ambos se minimiza en el punto óptimo  $d_{vc}^*$ .

*Observación 4.17.* El análisis basado en la dimensión VC depende únicamente del conjunto de hipótesis  $\mathcal{H}$ , sin considerar explícitamente el algoritmo de aprendizaje  $\mathcal{A}$ . En contraste, en el análisis de sesgo-varianza, el algoritmo de aprendizaje sí desempeña un papel fundamental en la descomposición del error, ya que, dado un mismo conjunto  $\mathcal{H}$ , distintos algoritmos pueden producir hipótesis diferentes.

No obstante, el análisis VC proporciona una intuición similar a la obtenida en el análisis de sesgo-varianza, en cuanto al comportamiento del error de generalización a medida que aumenta el tamaño de la muestra. El error de generalización puede acotarse mediante una cota que depende tanto del error empírico como de un término de complejidad vinculado a la dimensión VC del conjunto de hipótesis. Así, si la función de crecimiento es finita, el segundo término decrece con el tamaño de la muestra.

Otra forma clásica de caracterizar la capacidad de generalización de un modelo es mediante la medición de su capacidad para ajustarse a datos aleatorios. Este enfoque proporciona una indicación de su capacidad para generalizar a datos no observados.

**Definición 4.18** (Complejidad de Rademacher). La **complejidad de Rademacher** de un espacio de hipótesis  $\mathcal{H}$  y una muestra de ejemplos  $\{x_1, x_2, \dots, x_n\}$ , extraída del conjunto de

#### 4. El Dilema Clásico del Aprendizaje

entrenamiento  $\mathcal{D}$ , se define como:

$$\mathcal{R}(\mathcal{H}) = \frac{1}{n} \mathbb{E}_\sigma \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^n \sigma_i h(x_i) \right],$$

donde  $\sigma_i$  son variables aleatorias de Rademacher (véase Subsección 1.3.2), es decir, son variables (*i.i.d.*) que toman valores en el conjunto  $\{-1, 1\}$  con igual probabilidad.

De esta forma, la complejidad de Rademacher mide la capacidad de un modelo para ajustarse al ruido aleatorio uniforme generado por variables de Rademacher. Esta medida es fundamental porque nos ayuda a proporcionar una justificación teórica para la minimización del riesgo empírico (ERM), como se puede ver en el siguiente resultado.

**Teorema 4.19.** *El riesgo real de una hipótesis  $g \in \mathcal{H}$  está acotado por:*

$$L(g) \leq L_{emp}(g) + 2\mathcal{R}(\mathcal{H}) + C,$$

donde  $C$  es una constante que depende de la función de pérdida, el número de ejemplos, y la confianza  $1 - \delta$  del límite.

Esta acotación nos indica que si el espacio de hipótesis es muy flexible (es decir, contiene muchas hipótesis posibles), el límite es más laxo y poco informativo, lo que no proporciona garantías fuertes sobre la capacidad de generalización del modelo.

En conclusión, la dimensión VC y la complejidad de Rademacher, aunque son dos conceptos ciertamente distintos, están estrechamente relacionados y forman parte del marco teórico clásico del aprendizaje. Ambos están orientados a entender cómo un modelo puede ajustarse a los datos de entrenamiento y, al mismo tiempo, generalizar a nuevos datos, proporcionándonos acotaciones sobre el error de generalización. Además, ambas medidas dependen del espacio de hipótesis  $\mathcal{H}$ , pero no del algoritmo de aprendizaje específico que elija una hipótesis dentro de  $\mathcal{H}$ , siendo sus acotaciones menos efectivas a medida que dicho conjunto aumenta.

## **Parte II.**

### **Estado del Arte**



## 5. Trabajos Relacionados

El *Deep Double Descent*, al estar estrechamente relacionado al avance tecnológico y al crecimiento en la capacidad y el tamaño de los modelos de aprendizaje profundo, ha despertado un mayor interés en los últimos años, como puede observarse en la Figura 5.1. No obstante, aunque se percibe durante los últimos años una tendencia al alza del número de artículos que hacen referencia al mismo, únicamente encontramos un total 204 publicaciones en Scopus<sup>1</sup>. Esta reciente relevancia ilustra el carácter **innovador y pionero** de este proyecto.

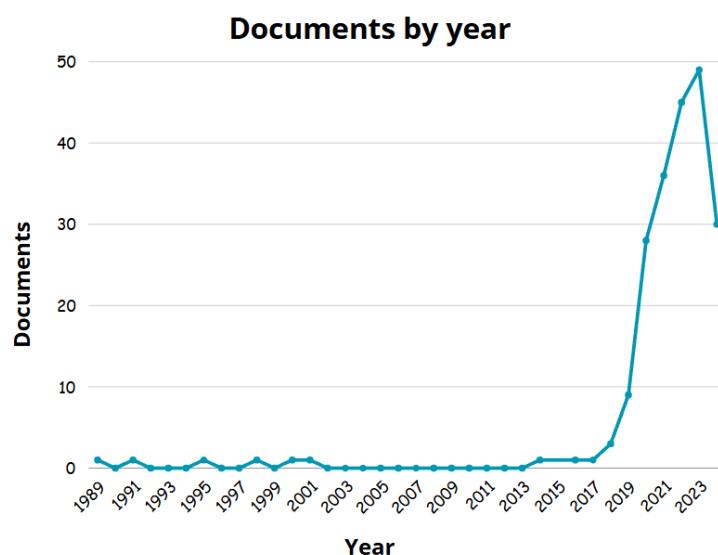


Figura 5.1.: Número de publicaciones relativas al *Deep Double Descent* en función del año de publicación. Se utilizan las referencias recuperadas desde Scopus<sup>1</sup>, complementadas con aquellas que fueron excluidas durante el proceso de filtrado o que no se encuentran disponibles en dicha base de datos.

Sin embargo, aunque la cantidad de artículos científicos aún sea limitada, el interés por parte de investigadores y científicos está creciendo rápidamente. Incluso en ausencia de publicaciones científicas formales, se continúan obteniendo nuevos resultados, tanto teóricos como prácticos, que continúan enriqueciendo nuestra comprensión del problema.

<sup>1</sup>Encontradas 204 publicaciones a fecha 12 de abril de 2025 usando la consulta: TITLE-ABS-KEY ((deep AND double AND descent) OR (overparameterized AND generalization)) AND PUBYEAR < 2025 AND (LIMIT-TO (SUBJAREA, "COMP") OR LIMIT-TO (SUBJAREA, "ENGI") OR LIMIT-TO (SUBJAREA, "MATH") OR LIMIT-TO (SUBJAREA, "PHYS")) AND (LIMIT-TO (EXACTKEYWORD, "Machine Learning") OR LIMIT-TO (EXACTKEYWORD, "Deep Learning") OR LIMIT-TO (EXACTKEYWORD, "Generalization") OR LIMIT-TO (EXACTKEYWORD, "Performance") OR LIMIT-TO (EXACTKEYWORD, "Neural-networks") OR LIMIT-TO (EXACTKEYWORD, "Deep Neural Networks") OR LIMIT-TO (EXACTKEYWORD, "Overparameterization") OR LIMIT-TO (EXACTKEYWORD, "Overfitting") OR LIMIT-TO (EXACTKEYWORD, "Generalization Error") OR LIMIT-TO (EXACTKEYWORD, "Generalization Performance") OR LIMIT-TO (EXACTKEYWORD, "Neural Networks") OR LIMIT-TO (EXACTKEYWORD, "Interpolation") OR LIMIT-TO (EXACTKEYWORD, "Generalize")).

## 5. Trabajos Relacionados

A pesar de la disponibilidad de artículos que abordan el tema, la mayoría de ellos no proporcionan una explicación detallada del mismo, centrándose generalmente en casos prácticos y dejando de lado el análisis teórico intrínseco, limitando su comprensión completa.

### 5.1. Origen y primeras manifestaciones

La primera publicación formal relacionada con el *Deep Double Descent* se remonta al año 1989, como se observa en la Figura 5.1. En ese año, Vallet et al. [VCR89] presentaron, de manera empírica, esta dinámica al emplear la pseudoinversa para abordar problemas de regresión lineal con datos artificiales. La siguiente evidencia empírica fue presentada por Krogh & Hertz en el año 1991 [KH91], quienes, de manera parcial, mostraron su aparición utilizando un modelo de regresión lineal.

Posteriormente, nos remontamos hasta 1995 cuando Opper presenta los primeros resultados teóricos en su artículo “Statistical Mechanics of Generalization” [Opp95], a través del uso de una red neuronal formada por un perceptrón de una sola capa con una función de activación lineal conocida como ADALINE [WH88], y, más tarde, en su revisión del artículo en 2001 “Learning to Generalize” [Opp01].

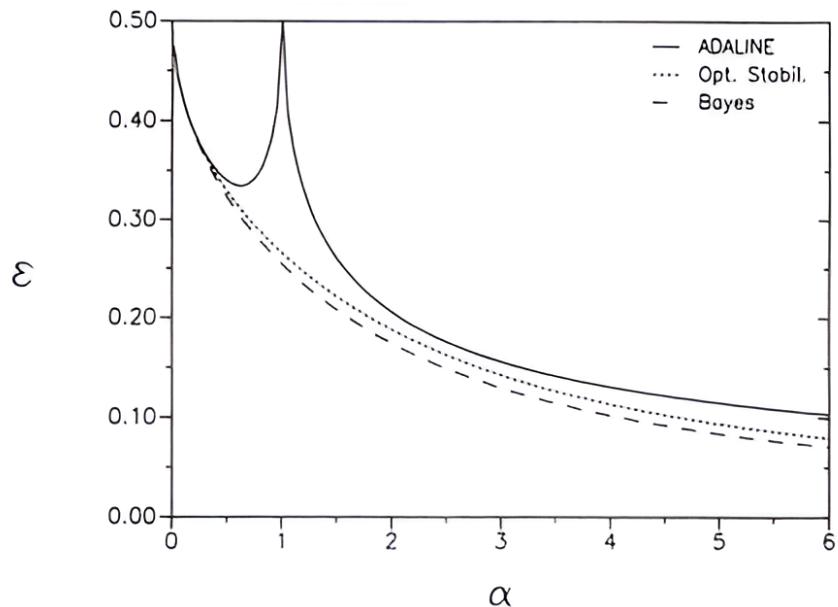


Figura 5.2.: Comparación del error de generalización ( $\epsilon$ ) para distintos modelos en función de la fracción entre el número de ejemplos aprendido y el número de parámetros ( $\alpha$ ). Se observa la curva del doble descenso para el modelo ADALINE [Opp95].

En sus representaciones (véase Figura 5.2), el error de generalización ( $\epsilon$ ) se muestra en función del número de ejemplos de entrenamiento ( $n$ ) y el número de parámetros ( $P$ ), donde  $\alpha = \frac{n}{P}$ . Se observa que el error de generalización alcanza su máximo cuando  $\alpha$  se aproxima a 1, es decir, cuando el número de ejemplos de entrenamiento es similar al número de parámetros del modelo. Sumado a ello, demuestra que, para ciertas configuraciones, cuando  $N$  tiende a infinito, la solución que proporciona la pseudoinversa mejora a medida que nos alejamos del máximo ( $\alpha = 1$ ).

Comportamientos similares a los obtenidos por Opper también han sido reportados por Advani & Saxe en 2017 [AS17], así como por Spigler et al. y Geiger et al. en 2019 [SGd<sup>+</sup>19, GSd<sup>+</sup>19]. Estos trabajos, anteriores a la formalización del fenómeno tal como se conoce hoy en día, trabajan con redes neuronales profundas con un gran número de parámetros y estudian el comportamiento del error de generalización utilizando herramientas de física estadística inspiradas en el trabajo de Opper. De esta manera, Spigler et al. y Geiger et al. proponen una conexión entre el *Deep Double Descent* y la transición *jamming* propia de la física estadística, mostrando por qué los modelos pueden llegar a generalizar mejor después del “pico” del error de test.

No obstante, fue Duin en el año 2000 [Du00] (véanse las Figuras 6 y 7 de su trabajo) el primero en mostrar curvas de generalización utilizando datos del mundo real, bastante similares a las curvas del doble descenso que tenemos hoy en día.

## 5.2. El nacimiento del *Deep Double Descent*

Iniciamos esta sección abordando el equilibrio clásico entre sesgo y varianza, un concepto fundamental en la teoría del aprendizaje automático [GBD92, HTF01, GB10]. Esta teoría sostiene que, a medida que la complejidad de un modelo aumenta, su sesgo disminuye, pero su varianza se incrementa. Como resultado, llega un punto en el que el error de generalización aumenta, formando la tradicional curva en forma de “U”. De acuerdo con esta visión tradicional, una vez superado cierto umbral de complejidad, los modelos más grandes son cada vez peores y, por tanto, se busca encontrar un equilibrio en el modelo. Sin embargo, los resultados prácticos modernos no comparten esta teoría. En la actualidad, la visión moderna entre los profesionales es que los «modelos grandes son mejores» [KSH12, NMB<sup>+</sup>19, HCB<sup>+</sup>19, SLJ<sup>+</sup>14]. Estos estudios muestran que el uso de redes neuronales con un gran número de parámetros conduce a un mejor rendimiento, evidenciando que los modelos más complejos pueden obtener resultados superiores a los modelos simples.

Este trabajo se enmarca dentro del estudio de la generalización en redes neuronales profundas, hecho por el que no ha sido percibido con claridad hasta hace relativamente poco tiempo, al estar ligado a una nueva tendencia que favorece el uso de modelos de gran tamaño, impulsada por el **incremento de la capacidad de cálculo disponible y la creciente disposición de grandes volúmenes de datos del mundo real**. En particular, se enlaza con investigaciones previas, como la de Zhang et al. en 2021 [ZBH<sup>+</sup>21], quienes argumentan que comprender el aprendizaje profundo aún requiere replantearse los paradigmas tradicionales de generalización, desafiando la noción clásica de sesgo-varianza.

El *Deep Double Descent* fue nombrado así, por primera vez, por Belkin et al. en 2019 [BHMM19], haciendo referencia a los dos descensos que presenta la curva del error de generalización. En este artículo, se busca unificar la teoría clásica del equilibrio sesgo-varianza con los resultados prácticos obtenidos por la teoría moderna, mediante una curva de error unificada (véase Figura 5.3). Belkin et al. muestran la aparición del doble descenso en diversos modelos y sobre distintos tipos de datos, entre los que se incluyen los árboles de decisión y redes neuronales poco profundas. Además, ofrece una primera intuición sobre su causa argumentando que, en la región sobreparametrizada, el modelo dispone de un mayor número de funciones candidatas compatibles con los datos. A esto se suma el efecto de la regularización implícita inducida por ciertos algoritmos de optimización, como el gradiente descendente [SHN<sup>+</sup>24].

Posteriormente, Nakkiran et al. en 2019 [NKB<sup>+</sup>19] observaron que el doble descenso no

## 5. Trabajos Relacionados

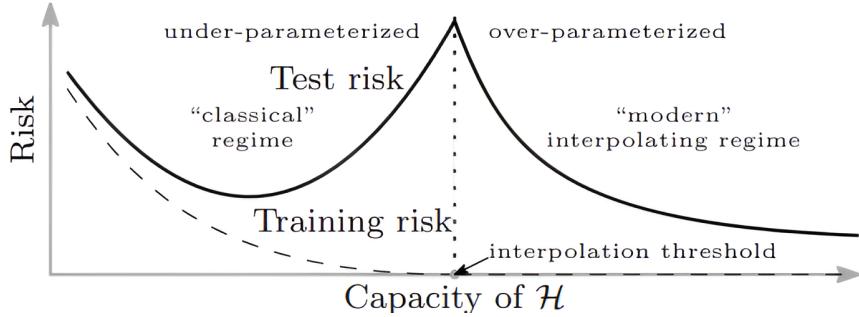


Figura 5.3.: Curvas para el error de entrenamiento (línea discontinua) y generalización (línea continua) [BHMM19]. Antes del umbral de interpolación (*under-parameterized*), aparece la curva clásica en "U". Tras dicho umbral (*over-parameterized*), se observa la curva del error moderna, induciendo el doble descenso.

solo dependía del tamaño del modelo, sino también del número de épocas de entrenamiento. Además, unificaron estos resultados mediante la introducción de una nueva medida de complejidad para un modelo: la **complejidad efectiva del modelo**, y conjeturaron bajo qué condiciones podía ocurrir en función de dicha medida. En este mismo trabajo, también formalizaron los distintos tipos que pueden manifestarse: en función del número de parámetros, del número de épocas y del tamaño del conjunto de entrenamiento, los cuales constituirán la base conceptual sobre la que desarrollaremos este proyecto.

A modo de reflexión final, es importante destacar que, aunque hoy en día el *Deep Double Descent* pueda parecer muy evidente, durante las últimas décadas la lógica predominante en la comunidad científica iba en otra dirección. En efecto, la mayoría de libros de texto y artículos científicos sobre redes neuronales profundas sostenían que debía controlarse el tamaño de los modelos para evitar el sobreajuste, siguiendo el paradigma clásico del equilibrio sesgo-varianza. Esta idea, que parecía una verdad absoluta e incuestionable, ha sido puesta en duda por estos nuevos comportamientos, motivados estrechamente por la enorme capacidad de cómputo de la que disponemos en la actualidad.

### 5.3. Avances recientes

En los últimos años, la comprensión del *Deep Double Descent* ha avanzado significativamente, gracias a nuevas investigaciones que han refinado su caracterización y explorado sus implicaciones en redes neuronales profundas. Uno de los principales avances en el campo del aprendizaje estadístico ha sido la reconsideración de los límites de la sabiduría clásica sobre el sesgo y la varianza, especialmente al analizar el impacto del uso de un gran número de parámetros en el aprendizaje [ZBH<sup>+</sup>21, CJvdS23]. Por otro lado, Schaeffer et al. en 2023 [SKR<sup>+</sup>23] realizaron los primeros estudios teóricos y experimentales enfocados en identificar y analizar las posibles causas y factores que pueden desencadenar el fenómeno.

Otras líneas de investigación han explorado cómo ciertas técnicas pueden mitigar su presencia. Por ejemplo, Yang y Suzuki en 2023 [YS24] analizaron el impacto de la regularización mediante el uso de *dropout*, demostrando que dicha técnica puede reducir la magnitud del segundo descenso en el error de generalización. Asimismo, Heckel y Yilmaz en 2021 [HY20] investigaron cómo el uso de la parada anticipada o *early stopping* puede influir en su aparición.

Desde una perspectiva más empírica, varios estudios han analizado su manifestación en escenarios de aprendizaje adversario. En particular, Min et al. en 2021 [MCK20] demostraron que, en algunos casos, un aumento de los datos de entrenamiento puede mejorar la robustez del modelo, aunque también puede provocar un descenso adverso en la generalización.

Singh et al. en 2022 [SLHS22] presentaron un análisis teórico en redes neuronales de tamaño finito, proporcionando una caracterización matemática básica que ayuda a entender los mecanismos subyacentes. Somepalli et al. en 2022 [SFB<sup>+</sup>22] investigaron la reproducibilidad del aprendizaje en redes neuronales y su relación con el *Deep Double Descent*.

Por otra parte, estudios modernos han abordado este comportamiento desde la perspectiva de los sesgos inductivos en redes neuronales. A principios de este año (2025), Andrew Wilson [Wil25] plantea que la generalización puede entenderse intuitivamente y formalizarse mediante marcos teóricos consolidados, introduciendo los **sesgos inductivos suaves** como principio unificador. Esta visión coincide con la de Mingard et al. (2023) [MRVPL23], quienes argumentan que las redes neuronales siguen un principio similar al de la navaja de Ockham.

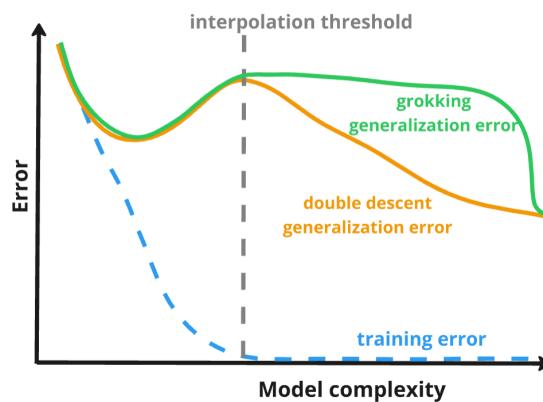


Figura 5.4.: Error en entrenamiento y test para el *grokking* (línea verde) y el doble descenso (línea naranja). Se observa que el *grokking* implica una mejora abrupta en términos de generalización una vez superado el umbral de interpolación, mientras que el doble descenso muestra una mejora progresiva a partir de ese punto.

Finalmente, investigaciones recientes han mostrado extensiones del doble descenso, pues este no está necesariamente limitado a dos descensos, sino que, bajo ciertas circunstancias, pueden observarse más de dos descensos [dSB21, CMBK21]. Además, se ha estudiado su relación con otros sucesos emergentes en el aprendizaje profundo, como es el caso del *grokking* [PBE<sup>+</sup>22]. En este contexto, Davies et al. en 2023 [DLK23] propusieron una conexión entre ambos, sugiriendo que el aprendizaje prolongado puede llevar a una mejora abrupta de la generalización (véase Figura 5.4). El *grokking*, en particular, suele asociarse a conjuntos de datos pequeños y estructurados, tras un entrenamiento prolongado en el que el modelo transita abruptamente del sobreajuste a la generalización. Por su parte, el DDD se manifiesta típicamente en modelos de gran capacidad entrenados con datos ruidosos, en los que se refleja una tendencia decreciente del error vinculada a la capacidad del modelo.

Para concluir este capítulo, en la Tabla 5.1 se presenta un resumen de las principales publicaciones relacionadas con el *Deep Double Descent*, junto con sus contribuciones más relevantes y el número de citas, obtenidas a partir de Google Scholar<sup>2</sup>.

<sup>2</sup>Número de citas encontradas a fecha 12 de abril de 2025.

## 5. Trabajos Relacionados

Año	Referencia	Principales contribuciones	Citas
1989	<i>Linear and nonlinear extension of the pseudoinverse solution for learning boolean functions.</i> Vallet et al. [VCR89]	Primera manifestación empírica sobre datos artificiales en la física teórica.	71
1991	<i>A simple weight decay can improve generalization.</i> Krogh & Hertz [KH91]	Evidenciaron, parcialmente, el fenómeno en el contexto de la regresión lineal.	2654
1995	<i>Statistical mechanics of learning: Generalization.</i> Manfred Opper [Opp95]	Primeros resultados teóricos formales sobre datos artificiales.	71
2000	<i>Classifiers in almost empty spaces.</i> Robert Duin [Du00]	Primeras evidencias empíricas usando datos del mundo real.	209
2019	<i>Reconciling modern machine learning practice and the classical bias-variance trade-off.</i> Belkin et al. [BHMM19]	Unificaron la sabiduría clásica con los enfoques modernos y acuñaron el nombre de <i>Deep Double Descent</i> .	2331
2019	<i>Deep double descent: Where bigger models and more data hurt.</i> Nakkiran et al. [NKB <sup>+</sup> 19]	Introdujeron la complejidad efectiva del modelo y los distintos tipos de doble descenso.	1174
2021	<i>Multiple descent: Design your own generalization curve.</i> Chen et al. [CMBK21]	Demostraron que el error puede presentar un número arbitrario de “picos” y que pueden controlarse.	80
2023	<i>Double descent demystified: Identifying interpreting &amp; ablating the sources of a deep learning puzzle.</i> Schaeffer et al. [SKR <sup>+</sup> 23]	Analizaron los factores que provocan su aparición en modelos de regresión polinómica.	29
2023	<i>Do deep neural networks have an inbuilt occam's razor?</i> Mingard et al. [MRVPL23]	Proponen que las redes neuronales trabajan siguiendo la filosofía de Ockham.	13
2023	<i>Unifying grokking and double descent.</i> Davies et al. [DLK23]	Plantean unificar el <i>grokking</i> y el <i>double descent</i> bajo un marco conceptual común.	37

Tabla 5.1.: Resumen de los principales artículos junto con sus contribuciones y citas, extraídas de Google Scholar<sup>2</sup>.

## **Parte III.**

### **Análisis Teórico y Empírico**



## 6. Análisis Teórico del Deep Double Descent

En este capítulo nos vamos a encargar de abordar, de manera teórica, el *Deep Double Descent*. En primer lugar, lo definiremos con la mayor precisión posible, ofreciendo una intuición clara a partir de un problema de regresión. A continuación, se presentarán los principales desarrollos presentes en la literatura científica, así como algunos avances recientes en el tema. Para concluir el capítulo, se abordará la teoría de la aproximación no lineal, ya que, a priori, presenta ciertas analogías con este fenómeno.

### 6.1. Planteamiento teórico

Siguiendo los resultados expuestos en la Sección 4.4, la sabiduría clásica adopta una visión en la cual sostiene que los modelos más grandes tienden a ser peores, ya que su capacidad de generalización empeora. En la práctica moderna, especialmente ahora en la era del *deep learning*, es cada vez más común el uso de modelos de gran tamaño con suficientes parámetros para reducir el error de entrenamiento casi a cero. A pesar de ajustarse casi perfectamente a los datos, estos modelos consiguen generalizar sorprendentemente bien, e incluso superar en rendimiento a modelos más simples.

De esta manera, se ha comprobado que, más allá de cierto umbral, el aumento de la capacidad de los modelos resulta beneficioso, ya que no conduce al sobreajuste y, en realidad, disminuye nuevamente el error de generalización. A esta nueva zona de funcionamiento de los modelos la denotaremos como **régimen moderno o zona sobreparametrizada**, mientras que la región previa al umbral, en la que se produce la tradicional curva con forma de "U", la denominaremos como **régimen clásico o zona infraparametrizada**.

De cara a formalizar el fenómeno del DDD y unificar la sabiduría clásica con la práctica moderna, introducimos una nueva medida de capacidad del modelo, propuesta por Nakkiran et al. en [NKB<sup>+</sup>19].

**Definición 6.1** (Complejidad efectiva del modelo). La **complejidad efectiva del modelo** (EMC, por sus siglas en inglés) de un algoritmo de aprendizaje  $\mathcal{A}$ , con respecto a la distribución de probabilidad conjunta  $P[X, Y]$  de los datos del conjunto de entrenamiento  $\mathcal{D}$ , es el máximo número de ejemplos de entrenamiento ( $n$ ) en el que  $\mathcal{A}$  obtiene, de media, un error de entrenamiento muy próximo a cero. Es decir, dado  $\epsilon > 0$ :

$$EMC_{P,\epsilon}(\mathcal{A}) = \max\{n \in \mathbb{N} \mid E_{in}[\mathcal{L}(g)] \leq \epsilon\},$$

donde  $\mathcal{L}(g)$  hace referencia a la función de pérdida de la función candidata  $g \in \mathcal{H}$ .

Una vez definida la noción de complejidad efectiva del modelo, se expone uno de los resultados principales de este trabajo.

**Hipótesis 6.2.** *Para cualquier distribución de datos  $P[X, Y]$ , algoritmo de aprendizaje basado en redes neuronales  $\mathcal{A}$  (véase Subsección 4.1.1.2) y un pequeño  $\epsilon > 0$ , si consideramos la tarea de predecir etiquetas basadas en  $n$  muestras aleatorias e independientes de  $P[X, Y]$ , se verifica:*

## 6. Análisis Teórico del Deep Double Descent

- **Región infraparametrizada.** Si  $\text{EMC}_{P,\epsilon}(\mathcal{A})$  es suficientemente menor que  $n$ , entonces cualquier perturbación de  $\mathcal{A}$  que aumente su complejidad efectiva, disminuirá su error de generalización.
- **Región crítica.** Si  $\text{EMC}_{P,\epsilon}(\mathcal{A}) \approx n$ , entonces cualquier perturbación de  $\mathcal{A}$  que aumente su complejidad efectiva puede aumentar o disminuir su error de generalización.
- **Región sobreparametrizada.** Si  $\text{EMC}_{P,\epsilon}(\mathcal{A})$  es suficientemente mayor que  $n$ , entonces cualquier perturbación de  $\mathcal{A}$  que aumente su complejidad efectiva, disminuirá su error de generalización.

Esta hipótesis es informal en varios sentidos. En primer lugar, no disponemos de una forma precisa de elegir el parámetro  $\epsilon$ , ni de una especificación formal para “suficientemente pequeño” y “suficientemente grande”. La hipótesis sugiere que hay un intervalo crítico alrededor del umbral de interpolación ( $\text{EMC}_{P,\epsilon}(\mathcal{A}) = n$ ) de manera que, tanto por debajo como por encima de dicho intervalo, el aumento de la complejidad beneficia el rendimiento del modelo, mientras que en el interior de este intervalo el comportamiento es incierto, pudiendo mejorar o empeorar. Además, la amplitud de dicho intervalo depende tanto de la distribución de los datos del conjunto de entrenamiento  $\mathcal{D}$  como del algoritmo de aprendizaje  $\mathcal{A}$  utilizado.

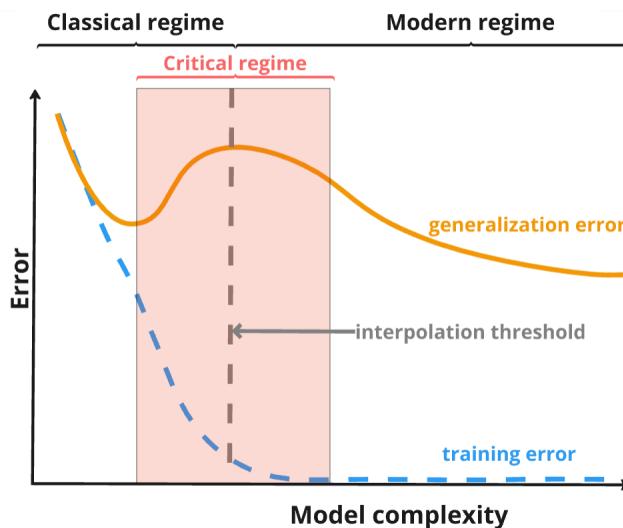


Figura 6.1.: Ejemplo de doble descenso con las distintas zonas de parametrización. Se puede observar la tradicional forma de “U” de la curva del error de generalización en la zona clásica (*Classical regime*), seguida de la zona crítica (*Critical regime*), donde el error de generalización es incierto, ya que puede aumentar o disminuir, y, finalmente, la zona moderna (*Modern regime*), donde el error de generalización es incluso menor que el obtenido en la zona clásica.

Por otra parte, la Hipótesis 6.2 nos ayuda a unificar la sabiduría clásica con los resultados modernos en una curva de aprendizaje que abarca ambos mundos. Hasta el umbral de interpolación, la curva sigue la tradicional forma de “U”, cuyo mínimo se alcanza en el *sweet spot* (véase Figura 4.7) mientras que, aumentando la capacidad del modelo hasta alcanzar un error de entrenamiento muy próximo a cero, y tras superar la zona crítica donde se encuentra

el umbral de interpolación, nos indica que el error de generalización comienza nuevamente a disminuir, pudiendo lograr un error menor que el obtenido en el *sweet spot*.

Este comportamiento, en el que la curva del error de generalización exhibe dos descensos, puede apreciarse en la Figura 6.1 y lo denotaremos como **doble descenso profundo**, o simplemente **doble descenso**. En este contexto, el adjetivo profundo hace referencia al uso de redes neuronales profundas como modelo base para el aprendizaje.

Aun habiendo unificado ambos puntos de vista, la descomposición del error en términos de sesgo y varianza, tal como se mencionó en la Sección 4.2, sugiere que uno de estos dos componentes, tras alcanzar el valor máximo de error, comienza nuevamente a disminuir. La Figura 6.2 ilustra los tres comportamientos distintos que pueden surgir como resultado de dicha descomposición. Para nuestro análisis, resulta razonable suponer que es la varianza la que disminuye tras alcanzar el punto máximo del error, ya que sabemos que este término predomina a partir del *sweet spot*, dando lugar a una curva unimodal.

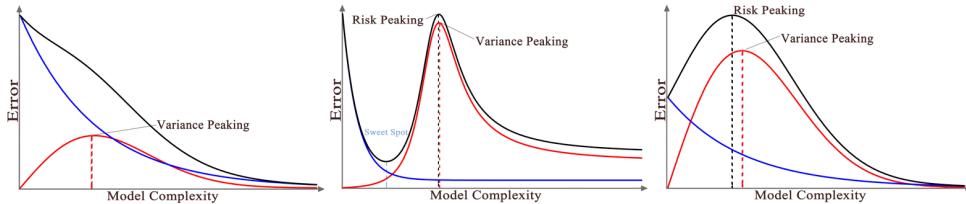


Figura 6.2.: Curvas típicas del error (línea negra) en función del sesgo (línea azul) y la varianza (línea roja) [YYY+20]. A la izquierda, el término de sesgo domina completamente, lo que da lugar a un error decreciente. En la imagen central, el sesgo y la varianza predominan en zonas distintas, produciéndose así el doble descenso. Finalmente, a la derecha, el término de varianza se vuelve dominante, generando una curva con forma de campana en la que el error inicialmente aumenta.

De este modo, a medida que aumenta la capacidad del modelo, esto implica que, a pesar de disponer de un conjunto de hipótesis más amplio, el modelo se está concentrando en un subconjunto específico de dichas hipótesis, lo que reduce el error asociado a la varianza.

A modo de resumen, estamos separando las distintas zonas de funcionamiento de un modelo en función, en cierta medida, de su conjunto de hipótesis  $\mathcal{H}$ . En la zona clásica, por lo general, no existe ninguna hipótesis que minimice completamente el riesgo real, es decir, no existe  $g \in \mathcal{H}$  tal que  $\mathcal{L}(g) = 0$ . Por el contrario, en la zona moderna, conforme aumenta la capacidad del modelo, aparece un conjunto cada vez más amplio  $S \subset \mathcal{H}$  que interpola a la perfección los datos de entrenamiento. Este conjunto se define como:

$$S = \{g \in \mathcal{H} \mid \mathcal{L}(g) = 0\}.$$

En consecuencia, el verdadero problema radica en comprender por qué el algoritmo de aprendizaje  $\mathcal{A}$ , en la zona sobreparametrizada, tiende a seleccionar “buenas” soluciones dentro del conjunto  $S$ , es decir, aquellas que logran una buena generalización. Es importante destacar que esta cuestión no se puede responder únicamente a partir de los datos del conjunto de entrenamiento  $\mathcal{D}$ , ya que cualquier hipótesis de  $S$  se ajusta perfectamente a esos datos. Por tanto, la clave para comprender el fenómeno vendrá dada por el sesgo inductivo que presentan algunos de los algoritmos de optimización más comunes, los cuales favorecen ciertas soluciones dentro de  $S$ .

### 6.1.1. Análisis intuitivo en un problema de mínimos cuadrados

De cara a ofrecer una primera intuición sencilla sobre la ocurrencia del DDD, consideramos un problema de regresión lineal (basado en [SKR<sup>+23</sup>]) que resolveremos utilizando el método de mínimos cuadrados ordinarios (OLS), cuya solución de norma mínima viene dada por la pseudoinversa, como vimos en la Sección 2.2. Además, para comprender dónde y cómo se produce este suceso, estudiaremos las dos zonas de parametrización del modelo de regresión lineal, analizando los diferentes errores que se producen en cada una de ellas.

Supongamos que nos enfrentamos a un problema de regresión lineal en el que disponemos de  $N$  ejemplos, donde cada ejemplo está formado por los respectivos datos de entrada  $x$  y su correspondiente etiqueta. Sea  $D$  la dimensión de los puntos de entrenamiento y  $P$  el número de parámetros a ajustar para realizar la regresión lineal. De esta manera, nuestro conjunto de entrenamiento  $\mathcal{D}$  está formado por  $N$  pares de la forma  $(x, y)$ , donde  $x \in \mathbb{R}^D$  representa un punto en el plano  $D$ -dimensional e  $y \in \mathbb{R}$  es la variable objetivo.

Para resolver el problema de regresión lineal, debemos abordar el clásico problema de minimización de mínimos cuadrados, formulado como sigue

$$\arg \min_w \frac{1}{N} \sum_{n=1}^N \|x_n \cdot w - y_n\|^2 = \arg \min_w \|Xw - Y\|^2, \quad (6.1)$$

donde cada par  $(x, y)$  representa un ejemplo de entrenamiento y  $w$  indica el vector de pesos o parámetros que queremos aprender de cara a realizar la aproximación. Además,  $X \in \mathcal{M}_{N \times D}(\mathbb{R})$  y  $Y \in \mathcal{M}_{N \times 1}(\mathbb{R})$  representan, respectivamente, los puntos de entrenamiento y su correspondiente salida en forma matricial.

En la zona infraparametrizada, es decir, cuando el número de ejemplos de entrenamiento es mayor que el número de parámetros a ajustar ( $P < N$ ), la solución al problema de minimización (6.1), ofrecida por la pseudoinversa de  $X$ , viene dada por  $(X^T X)^{-1} X^T Y$ . De este modo, el vector de pesos  $w$  puede ser expresado como:

$$w_{under} = (X^T X)^{-1} X^T Y.$$

Por otra parte, en la zona sobreparametrizada, es decir, cuando el número de parámetros es mayor que el número de ejemplos de entrenamiento ( $N < P$ ), no se puede resolver el problema de minimización (6.1) de manera convencional, dado que estaría mal planteado, debido al hecho de que existirían múltiples soluciones, puesto que habría menos restricciones que parámetros. Por tanto, debemos seleccionar un problema de optimización distinto, que también se encuentre sujeto a las restricciones impuestas por los ejemplos de entrenamiento. En este contexto, elegimos el problema más simple que se podría imaginar, dado por:

$$\arg \min_w \|w\|^2, \quad \text{sujeto a } x_n \cdot w = y_n \quad \forall n \in \{1, \dots, N\}. \quad (6.2)$$

El problema de optimización (6.2) busca el vector de parámetros con menor norma que satisface  $x_n \cdot w = y_n$  para todos los ejemplos de entrenamiento. La solución a este problema de optimización también se obtiene mediante la pseudoinversa, pero en este caso, la pseudoinversa se calcula knowing que la matriz  $X$  tiene rango completo en esta región y cuenta con filas linealmente independientes. Así, la pseudoinversa viene dada por  $X^T (X X^T)^{-1} Y$  (véase Sección 2.2). De esta forma, el vector  $w$  en la zona sobreparametrizada viene dado por:

$$w_{over} = X^T (X X^T)^{-1} Y.$$

Por tanto, una vez que se ha obtenido el vector de pesos, el modelo realizará las siguientes predicciones para un determinado punto de prueba  $x_{\text{test}}$ , dependiendo de la zona de parametrización en la que se encuentre:

$$y_{\text{test},\text{under}} = x_{\text{test}} \cdot w_{\text{under}} = x_{\text{test}} \cdot (X^T X)^{-1} X^T Y.$$

$$y_{\text{test},\text{over}} = x_{\text{test}} \cdot w_{\text{over}} = x_{\text{test}} \cdot X^T (X X^T)^{-1} Y.$$

No obstante y llegados a este punto, las diferencias al predecir un punto de prueba en ambas zonas de parametrización no parecen ser del todo claras. Para identificar estas diferencias con mayor precisión, reescribiremos nuestras predicciones de la siguiente forma:

$$y_n = x_n \cdot w^* + e_n,$$

donde  $w^* \in \mathbb{R}^P = \mathbb{R}^D$  representa el **vector de parámetros ideal** que realmente minimiza el error cuadrático medio, y  $e_n$  representa un término de error adicional presente en la propia naturaleza de los datos, es decir, un residuo que el modelo no puede capturar. Equivalentemente, en forma matricial, podemos escribir:

$$Y = X \cdot w^* + E, \quad \text{con } E \in \mathbb{R}^{N \times 1}.$$

Usando esta notación, podemos reformular las predicciones que realiza el modelo. De esta manera, para el caso de la zona infraparametrizada nos queda:

$$\begin{aligned} y_{\text{test},\text{under}} &= x_{\text{test}} \cdot (X^T X)^{-1} X^T Y \\ &= x_{\text{test}} \cdot (X^T X)^{-1} X^T (X w^* + E) \\ &= x_{\text{test}} \cdot (X^T X)^{-1} X^T X w^* + x_{\text{test}} \cdot (X^T X)^{-1} X^T E \\ &= \underbrace{x_{\text{test}} \cdot w^*}_{\stackrel{\text{def}}{=} y_{\text{test}}^*} + x_{\text{test}} \cdot (X^T X)^{-1} X^T E \end{aligned}$$

$$y_{\text{test},\text{under}} - y_{\text{test}}^* = x_{\text{test}} \cdot (X^T X)^{-1} X^T E.$$

Por otra parte, para el caso de la zona sobreparametrizada, los cálculos serían los siguientes:

$$\begin{aligned} y_{\text{test},\text{over}} &= x_{\text{test}} \cdot X^T (X X^T)^{-1} Y \\ &= x_{\text{test}} \cdot X^T (X X^T)^{-1} (X w^* + E) \\ &= x_{\text{test}} \cdot X^T (X X^T)^{-1} X w^* + x_{\text{test}} X^T (X X^T)^{-1} E \\ y_{\text{test},\text{over}} - \underbrace{x_{\text{test}} \cdot w^*}_{\stackrel{\text{def}}{=} y_{\text{test}}^*} &= x_{\text{test}} \cdot X^T (X X^T)^{-1} X w^* - x_{\text{test}} \cdot I_D w^* + x_{\text{test}} \cdot (X^T X)^{-1} X^T E \end{aligned}$$

$$y_{\text{test},\text{over}} - y_{\text{test}}^* = x_{\text{test}} \cdot (X^T (X X^T)^{-1} X - I_D) w^* + x_{\text{test}} \cdot (X^T X)^{-1} X^T E.$$

Las ecuaciones obtenidas son importantes, aunque, a simple vista, no nos proporcionan

## 6. Análisis Teórico del Deep Double Descent

información significativa. De cara a extraer intuiciones más claras, vamos a reemplazar la matriz  $X$  por su descomposición en valores singulares (véase Sección 2.2). De este modo, definimos  $X = U\Sigma V^T$ ,  $R = \text{rang}(X)$  y  $\sigma_1 > \dots > \sigma_R > 0$  como los valores singulares no nulos de  $X$ . Dado que  $E \in \mathbb{R}^{N \times 1}$ , podemos descomponer el error de predicción de la siguiente manera:

- En la zona infraparametrizada:

$$y_{test,under} - y_{test}^* = x_{test} \cdot V\Sigma^{\dagger}U^T E = \sum_{r=1}^R \frac{1}{\sigma_r} (x_{test} \cdot v_r)(u_r \cdot E),$$

donde se ha utilizado que  $V\Sigma^{\dagger}U^T$  corresponde a la descomposición en valores singulares de la pseudoinversa de  $X$ , dada por  $X^{\dagger} = (X^T X)^{-1} X^T$ .

- De manera análoga a la zona anterior, en la zona sobreparametrizada encontramos:

$$\begin{aligned} y_{test,over} - y_{test}^* &= x_{test} \cdot (X^T (XX^T)^{-1} X - I_D) w^* + x_{test} \cdot V\Sigma^{\dagger}U^T E \\ &= x_{test} \cdot (X^T (XX^T)^{-1} X - I_D) w^* + \sum_{r=1}^R \frac{1}{\sigma_r} (x_{test} \cdot v_r)(u_r \cdot E), \end{aligned}$$

donde se ha utilizado que  $V\Sigma^{\dagger}U^T$  corresponde a la descomposición en valores singulares de la pseudoinversa de  $X$ , dada por  $X^{\dagger} = X^T (XX^T)^{-1}$ .

Llegados a este punto, podemos analizar con mayor claridad las diferencias entre ambas predicciones. La zona sobreparametrizada incluye el término adicional  $x_{test} \cdot (X^T (XX^T)^{-1} X - I_D) w^*$ , que viene a ser una proyección del punto de test en el espacio de características. Recordemos que, en la zona sobreparametrizada, hay más parámetros que ejemplos de entrenamiento, por lo que, para  $N$  ejemplos de entrenamiento en  $D = P$  dimensiones, el modelo solo puede captar fluctuaciones de los datos en  $N$  dimensiones, pero no tiene visibilidad para captar las fluctuaciones en las restantes  $P - N$  dimensiones.

Esto provoca la pérdida de información sobre la relación lineal óptima del vector de pesos  $w^*$ , lo que a su vez aumenta el error de predicción. Este término es el asociado al sesgo (véase Sección 4.2), pues, al encontrarnos en la zona sobreparametrizada, disponemos de gran cantidad de hipótesis debido a la falta de suficientes restricciones en comparación con el número de parámetros. Esto favorece que la hipótesis ideal se encuentre dentro de ese conjunto de posibles soluciones, lo cual, a su vez, facilita que este término sea cercano a cero.

El otro término,  $\sum_{r=1}^R \frac{1}{\sigma_r} (x_{test} \cdot v_r)(u_r \cdot E)$ , representa la influencia de cada componente singular y es el causante del doble descenso en una regresión lineal resuelta mediante el método OLS. Este término se conoce como varianza y refleja el impacto de modelar las fluctuaciones de los datos en las direcciones singulares, y si dicho equilibrio está correlacionado con los objetivos de regresión.

Por tanto, estos tres términos, multiplicados, determinan cuánto contribuye la  $r$ -ésima componente singular al error de predicción. Además, el “pico” del primer ascenso ocurre cerca del umbral de interpolación, debido a que el menor valor singular no nulo suele alcanzar su valor más bajo en dicho umbral, basado en la distribución de Marchenko-Pastur [MP67].

A grandes rasgos, esta distribución describe el comportamiento de los valores propios de matrices aleatorias y se aplica cuando se tiene una matriz aleatoria cuyas entradas son

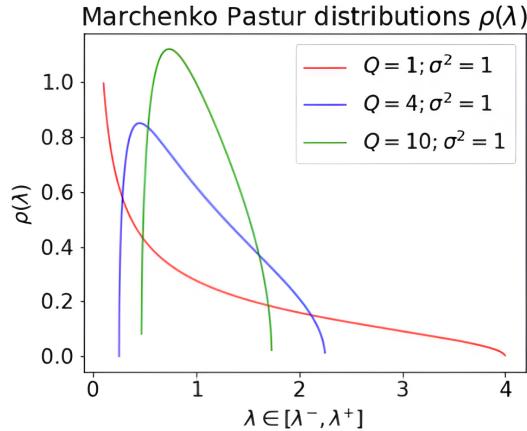


Figura 6.3.: Ejemplos de distribuciones de Marchenko-Pastur [MM18] para distintos valores de la relación de aspecto de la matriz ( $Q = \frac{N}{P}$ ) y varianza fija. Cuando  $Q = 1$ , la mayoría de valores propios se acumulan alrededor del cero. A medida que  $Q$  aumenta, los valores propios mínimos se alejan progresivamente de cero.

independientes e idénticamente distribuidas, como es el caso de la matriz  $X \in \mathcal{M}_{N \times D}(\mathbb{R})$ . En particular, la distribución de Marchenko-Pastur describe los valores propios de la matriz de covarianza  $XX^T$ , que a su vez determinan la varianza explicada por las componentes principales del modelo. De este modo, cuando la relación  $\frac{N}{P}$  se aproxima a 1, los valores propios más pequeños se acercan a 0 (véase Figura 6.3), provocando que los valores singulares también sean casi nulos. Esto ocurre cerca del umbral de interpolación, donde  $N = P = D$ , y para evitar que el  $N$ -ésimo dato añada un valor singular pequeño no nulo a los datos de entrenamiento, deben cumplirse dos condiciones:

- Debe haber una dimensión en la que ninguno de los datos de entrenamiento anteriores haya variado significativamente.
- El  $N$ -ésimo dato debe presentar una variación considerable en esa dimensión.

Sin embargo, este escenario es muy poco probable debido a que, en la mayoría de los casos, los datos de entrenamiento se muestran de manera independiente a partir de variables aleatorias que son i.i.d., lo que resulta en una distribución aleatoria de los datos en todas las dimensiones. Al superar este umbral de interpolación, la varianza explicada por cada dimensión de las covariables se hace más notoria, y los valores singulares no nulos más pequeños se alejan de cero, lo que reduce nuevamente el término de varianza en las predicciones, pudiendo provocar el doble descenso.

## 6.2. Convergencia del descenso de gradiente

En esta sección, de cara a profundizar en la intuición obtenida en la sección anterior, vamos a analizar la influencia de utilizar el descenso de gradiente como método de optimización en problemas de regresión y clasificación, dado que es uno de los métodos más utilizados para entrenar redes neuronales.

### 6.2.1. Problema de regresión

En primer lugar, nos centraremos en abordar un problema de regresión utilizando el descenso de gradiente (basándonos en [LT24]), en lugar de utilizar la solución que nos ofrece el problema de mínimos cuadrados asociado. Para ello, vamos a considerar nuevamente un problema de mínimos cuadrados bajo los mismos supuestos de la sección anterior. Por tanto, suponemos que nuestro conjunto de entrenamiento  $\mathcal{D}$  está formado por  $N$ -pares de la forma  $(x, y)$ , donde  $x \in \mathbb{R}^D$  e  $y \in \mathbb{R}$ . Así, nuestro conjunto de entrenamiento presenta la siguiente forma:

$$\mathcal{D} = \{(x_i, y_i) \in \mathbb{R}^D \times \mathbb{R}\}, \quad \text{con } i \in \{1, \dots, N\}.$$

Definimos  $X \in \mathcal{M}_{N \times D}(\mathbb{R})$  como la matriz cuyas filas son los vectores  $x_i^T$  e  $y \in \mathbb{R}^N$  como el vector columna cuyos elementos son los  $y_i$ . Recordemos que la regla de actualización del descenso de gradiente para el vector de parámetros  $w$ , utilizando la función de pérdida  $\mathcal{L}$  y una tasa de aprendizaje  $\eta$ , viene dada por:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla \mathcal{L}(\mathbf{w}).$$

De manera similar a la sección anterior, consideramos el siguiente problema de optimización para resolver la regresión lineal:

$$\min_{w \in \mathbb{R}^D} \mathcal{L}(w) = \min_{w \in \mathbb{R}^D} \frac{1}{2} \|Xw - y\|^2, \quad (6.3)$$

donde en la ecuación (6.3) se agrega el término  $\frac{1}{2}$  para simplificar los cálculos de las derivadas en pasos posteriores. De esta forma, podemos enfocarnos en analizar las propiedades de la solución que encuentra el descenso de gradiente.

**Teorema 6.3.** *El conjunto de soluciones de un problema de mínimos cuadrados, denotado por  $\mathcal{S}_{LS}$ , es exactamente el siguiente:*

$$\mathcal{S}_{LS} = \{X^\dagger y + (I_D - X^\dagger X)u, u \in \mathbb{R}^D\}.$$

*Demostración.* Podemos expresar el error de la ecuación (6.3) de la siguiente forma:  $Xw - y = Xw - XX^\dagger y + XX^\dagger y - y = Xw - XX^\dagger y - (I_D - XX^\dagger)y$ .

A partir de la última ecuación, sabemos que  $XX^\dagger y$  representa la proyección de  $y$  sobre el espacio columna de  $X$  y  $(I_D - XX^\dagger)y$  corresponde a su componente ortogonal. Esto se debe a que  $XX^\dagger y$  es la proyección sobre  $Im(X)$  y  $(I_D - XX^\dagger)y$  es la proyección en el complemento ortogonal de  $Im(X)$ , es decir, el  $\ker(X^T)$ . Para verificar estas afirmaciones, basta utilizar ambas propiedades de manera conjunta en el Lema 2.15.

Dado que ambos términos son ortogonales, podemos descomponer la norma del error de la siguiente forma:

$$\|Xw - y\|^2 = \|Xw - XX^\dagger y\|^2 + \|(I_D - XX^\dagger)y\|^2,$$

lo que nos lleva a la siguiente desigualdad:

$$\|Xw - y\|^2 \geq \|(I_D - XX^\dagger)y\|^2.$$

Finalmente, la igualdad se alcanza si y solo si:

$$Xw - XX^\dagger y = 0 \implies Xw = XX^\dagger y.$$

De esta manera, sabemos que  $X^\dagger y$  es una solución particular de la ecuación  $Xw = XX^\dagger y$ . Para obtener la solución general, basta con añadir cualquier vector que se encuentre en el núcleo de  $X$ , es decir, cualquier vector de la forma  $\{(I_D - X^\dagger X)u, u \in \mathbb{R}^D\}$  (véase Lema 2.15).  $\square$

Una vez que conocemos el conjunto de soluciones para nuestro problema de optimización, podemos analizar cómo se comporta dicho conjunto en función del rango de la matriz  $X$ .

*Observación 6.4.* Dependiendo del rango de la matriz  $X$ , el conjunto de soluciones  $\mathcal{S}_{LS}$  variará en función de la expresión de la pseudoinversa  $X^\dagger$ :

- Si  $N < D$  y  $\text{rang}(X) = N$ , entonces  $X^\dagger = X^T(XX^T)^{-1}$  y  $\mathcal{S}_{LS} = \{X^T(XX^T)^{-1}y + (I_D - X^T(XX^T)^{-1}X)u, u \in \mathbb{R}^D\}$ .
- Si  $D < N$  y  $\text{rang}(X) = D$ , entonces  $X^\dagger = (X^T X)^{-1} X^T$  y  $\mathcal{S}_{LS} = \{(X^T X)^{-1} X^T y\}$ .
- Si  $D = N$  y  $X$  tiene inversa, entonces  $X^\dagger = X^{-1}$  y  $\mathcal{S}_{LS} = \{X^{-1}y\}$ .

En particular, en los dos últimos casos, dado que  $\ker(X)$  es trivial, la solución del problema es única.

Por tanto, nos interesa centrarnos en analizar cómo de buena es la solución que proporciona el descenso de gradiente en la región sobreparametrizada, es decir, cuando  $N < D$ , dado que es en este caso cuando se generan múltiples soluciones, mientras que en los otros casos la solución es única.

**Teorema 6.5.** *Si el problema de mínimos cuadrados lineales (6.3) se encuentra en la región sobreparametrizada, es decir, ( $N < D$ ) y  $\text{rang}(X) = N$ , entonces, usando el descenso de gradiente con una tasa de aprendizaje constante  $0 < \eta < \frac{1}{\lambda_{\max}(X)}$ , donde  $\lambda_{\max}(X)$  es el mayor valor propio de  $X$ , y partiendo desde un punto inicial  $w_0 \in \text{Im}(X^T)$ , se garantiza la convergencia hacia la solución de norma mínima.*

*Demostración.* Dado que estamos suponiendo que  $X$  tiene  $N$  filas linealmente independientes, podemos expresar su descomposición en valores singulares de la siguiente manera:

$$X = U\Sigma V^T = [U_1 \quad U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix},$$

donde  $U \in \mathbb{R}^{N \times N}$  y  $V \in \mathbb{R}^{D \times D}$  son matrices ortogonales,  $\Sigma \in \mathbb{R}^{N \times D}$  es una matriz diagonal rectangular,  $U_1$  (respectivamente  $V_1$ ) son las submatrices que contienen los vectores singulares izquierdos (respectivamente derechos) asociados con los valores singulares no nulos y  $\Sigma_1 \in \mathbb{R}^{n \times n}$  es una matriz diagonal con valores singulares no nulos.

Nos centramos ahora en encontrar la solución de norma mínima  $w^*$ , la cual, como sabemos, viene dada por la pseudoinversa:

$$w^* = X^T(XX^T)^{-1}y.$$

Utilizando la descomposición en valores singulares de las matrices  $X$  y  $X^T$ , y dado que  $V$  es ortogonal, obtenemos:

## 6. Análisis Teórico del Deep Double Descent

$$XX^T = U\Sigma V^T V\Sigma^T U^T = U\Sigma\Sigma^T U^T.$$

Así, para la inversa de la matriz  $(XX^T)$ , sabiendo que  $U$  es ortogonal, nos queda:

$$(XX^T)^{-1} = U(\Sigma\Sigma^T)^{-1}U^T.$$

Volviendo a usar la descomposición en valores singulares para la matriz  $X^T$ ,  $X^T = V\Sigma U^T$ , obtenemos que la solución de norma mínima  $w^*$  se puede reescribir como sigue (los vectores singulares asociados a los valores singulares nulos no tienen impacto en la solución de mínimos cuadrados, pues su dirección es linealmente dependiente):

$$w^* = X^T(XX^T)^{-1}y = V\Sigma U^T U(\Sigma\Sigma^T)^{-1}U^T y = V_1\Sigma_1^{-1}U_1^T y.$$

A continuación, trabajamos sobre la regla de actualización del gradiente descendente, que viene dada por:

$$w^{(k+1)} = w^{(k)} - \eta \nabla \mathcal{L}(w) = w^{(k)} - \eta X^T(Xw^{(k)} - y) = (I - \eta X^T X)w^{(k)} + \eta X^T y,$$

donde  $X^T(Xw^{(k)} - y)$  hace referencia a la derivada de  $\mathcal{L}(w)$  respecto de  $w$ .

Seguidamente, aplicando inducción, obtenemos la expresión general:

$$w^{(k)} = (I - \eta X^T X)^k w_0 + \eta \sum_{l=0}^{k-1} (I - \eta X^T X)^l X^T y.$$

Ahora bien, usando la descomposición en valores singulares para la matriz  $X^T X$  ( $V\Sigma^T \Sigma V^T$ ) y dado que  $V$  es ortogonal, la iteración del gradiente descendente para el paso  $k$ -ésimo se expresa como:

$$w^{(k)} = V(I - \eta \Sigma^T \Sigma)^k V^T w_0 + \eta V \left( \sum_{l=0}^{k-1} (I - \eta \Sigma^T \Sigma)^l \Sigma^T \right) U^T y.$$

Reescribiendo la ecuación anterior en términos de  $\Sigma_1$ , obtenemos:

$$w^{(k)} = V \begin{bmatrix} (I - \eta \Sigma_1^2)^k & 0 \\ 0 & I \end{bmatrix} V^T w_0 + \eta V \left( \sum_{l=0}^{k-1} \begin{bmatrix} (I - \eta \Sigma_1^2)^l \Sigma_1 \\ 0 \end{bmatrix} \right) U^T y.$$

De cara a garantizar la convergencia del descenso de gradiente, elegimos  $0 < \eta < \frac{1}{\lambda_{\max}(\Sigma_1)}$ , lo que asegura que los valores propios de  $I - \eta \Sigma^T \Sigma$  sean estrictamente menores que 1, lo que implica, a su vez, que:

$$(I - \eta \Sigma_1^T \Sigma_1)^k \rightarrow 0 \quad \text{cuando} \quad k \rightarrow \infty,$$

y, por tanto,

$$V \begin{bmatrix} (I - \eta \Sigma_1^2)^k & 0 \\ 0 & I \end{bmatrix} V^T w_0 \xrightarrow{k \rightarrow \infty} V \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} V^T w_0 = V_2 V_2^T w_0,$$

dado que  $V_2^T w_0$  es la parte de  $w_0$  correspondiente a los valores singulares nulos de  $X^T X$ .

Además,

$$\eta \sum_{l=0}^{k-1} \begin{bmatrix} (I - \eta \Sigma_1^2)^l \Sigma_1 \\ 0 \end{bmatrix} \xrightarrow{k \rightarrow \infty} \eta \begin{bmatrix} \sum_{l=0}^{\infty} (I - \eta \Sigma_1^2)^l \Sigma_1 \\ 0 \end{bmatrix} = \begin{bmatrix} \eta(I - I + \eta \Sigma_1^2)^{-1} \Sigma_1 \\ 0 \end{bmatrix} = \begin{bmatrix} \Sigma_1^{-1} \\ 0 \end{bmatrix}.$$

Por consiguiente, denotando  $w_\infty$  como el límite de las iteraciones del gradiente descendente, obtenemos:

$$w_\infty = V_2 V_2^T w_0 + V_1 \Sigma_1^{-1} U^T y = V_2 V_2^T w_0 + X^T (X X^T)^{-1} y = V_2 V_2^T w_0 + w^*.$$

Finalmente, dado que  $w_0$  está en la imagen de  $X^T$ , podemos escribir  $w_0 = X^T z$  para algún  $z \in \mathbb{R}^N$ , lo que implica (usando la descomposición en valores singulares de la matriz  $X^T$ ):

$$V_2 V_2^T w_0 = V \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} V^T X^T z = V \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} V^T V \Sigma^T U^T z = V \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \Sigma_1 \\ 0 \end{bmatrix} U^T z = 0.$$

En consecuencia, el término  $V_2 V_2^T w_0$  desaparece, y nos queda que  $w_\infty = w^*$ , por lo que el gradiente descendente converge a la solución de norma mínima.  $\square$

En conclusión, tanto el gradiente descendente como la pseudoinversa llegan a la misma solución (bajo ciertas hipótesis) cuando se trata de resolver un problema de regresión lineal, ya que ambos buscan la solución interpolante de norma mínima, lo que los convierte en métodos equivalentes. De esta manera, **realizar el gradiente descendente hasta la convergencia, comenzando con una inicialización de todos los pesos a cero, es equivalente a aplicar directamente la pseudoinversa**. Por tanto, podemos interpretar que utilizar la pseudoinversa para resolver un problema de mínimos cuadrados es, en esencia, un caso particular del gradiente descendente aplicado al mismo problema.

### 6.2.2. Problema de clasificación con datos separables

En esta subsección nos preocupamos de verificar el efecto de utilizar el descenso de gradiente como método de optimización en un problema de clasificación binaria con un conjunto de datos linealmente separable [SHN<sup>+</sup>24]. Por simplicidad, consideraremos un problema de clasificación binaria con un conjunto de datos separable, lo que facilita la intuición sobre el comportamiento del descenso de gradiente. No obstante, los principios que se discuten posteriormente pueden extenderse a problemas de clasificación multiclase [RSSW24] y al uso del descenso de gradiente en redes neuronales [GLSS19].

**Definición 6.6.** Un conjunto de datos  $\mathcal{D} = \{(x_i, y_i) \mid i \in \{1, \dots, N\}\}$ , donde para todo  $i \in \{1, \dots, N\}$  se verifica que  $(x_i, y_i) \in \mathbb{R}^D \times \{-1, 1\}$ , es linealmente separable si existe  $w_* \in \mathbb{R}^D$  de manera que  $\forall i : y_i w_* x_i > 0$ .

Además, los resultados que se exponen en este apartado se cumplen para funciones de pérdida  $\ell : \mathbb{R} \rightarrow \mathbb{R}_+$  que presentan las siguientes propiedades:

1.  $\ell$  es positiva, diferenciable y decreciente de manera monótona a cero ( $\ell(u) > 0, \ell'(u) < 0$  y  $\lim_{u \rightarrow \infty} \ell(u) = \lim_{u \rightarrow \infty} \ell'(u) = 0$ ).
2.  $\ell$  es una función  $\beta$ -suave, es decir, el gradiente de  $\ell$  es  $\beta$ -Lipschitz ( $\forall u, v \in \mathbb{R}, \|\nabla \ell(u) - \nabla \ell(v)\| \leq \beta \|u - v\|$ ).

## 6. Análisis Teórico del Deep Double Descent

$$3. \lim_{u \rightarrow -\infty} \ell'(u) < 0.$$

Estas propiedades incluyen funciones de pérdida comunes para problemas de clasificación binaria, incluyendo la función logística y la exponencial. Además, estas propiedades implican que  $\mathcal{L}(w)$  (véase ecuación (6.4)) es una función  $\beta\sigma_{\max}^2(X)$ -suave (ligado al hecho de que  $\ell$  es  $\beta$ -suave), donde  $\sigma_{\max}(X)$  es el máximo valor singular de la matriz  $X \in \mathbb{R}^{D \times N}$  que contiene los datos de entrenamiento.

Por tanto, para nuestro problema, consideramos un conjunto de entrenamiento  $\mathcal{D}$  linealmente separable formado por  $N$ -pares de datos de la forma  $(x_i, y_i)$  con  $x_i \in \mathbb{R}^D$  e  $y_i \in \{-1, 1\}$ , donde  $i \in \{1, \dots, N\}$ , y nos centramos en minimizar la función de pérdida dada por:

$$\min_{w \in \mathbb{R}^D} \mathcal{L}(w) = \min_{w \in \mathbb{R}^D} \sum_{i=1}^N \ell(y_i w^T x_i), \quad (6.4)$$

donde  $w \in \mathbb{R}^D$  es el vector de parámetros y  $\ell$  es una función de pérdida binaria verificando las propiedades 1, 2 y 3 descritas anteriormente.

Seguidamente, nos ocupamos de estudiar la solución que ofrece el descenso de gradiente para este problema, fijada una tasa de aprendizaje  $\eta$ :

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla \mathcal{L}(\mathbf{w}) = \mathbf{w}^{(\tau)} - \eta \sum_{i=1}^N \ell'(y_i w^T x_i) y_i x_i.$$

De cara a estudiar esta solución, utilizaremos un lema auxiliar que nos ayudará con la demostración del resultado principal de esta subsección.

**Lema 6.7.** *Sea  $\mathcal{L}(w)$  una función  $\beta$ -suave no negativa. Si  $\eta < \frac{2}{\beta}$ , entonces, para cualquier  $w_0$  y utilizando el método del descenso de gradiente dado por:*

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla \mathcal{L}(w),$$

*se tiene que  $\sum_{u=0}^{\infty} \|\nabla \mathcal{L}(w^{(u)})\|^2 < \infty$  y, por tanto:*

$$\lim_{t \rightarrow \infty} \|\nabla \mathcal{L}(w^{(t)})\|^2 = 0.$$

La demostración del lema se puede verificar en el Apéndice A.

Una vez conocemos el resultado anterior, podemos enunciar el principal teorema de esta subsección, que establece las condiciones necesarias para que el descenso de gradiente converja hacia un punto crítico (minimizador global).

**Teorema 6.8.** *Sea  $\mathcal{D}$  un conjunto de entrenamiento linealmente separable con  $N$  ejemplos y  $\ell : \mathbb{R} \rightarrow \mathbb{R}_+$  una función de pérdida verificando las propiedades 1, 2 y 3. Sean  $w_t$  las iteraciones del descenso de gradiente usando una tasa de aprendizaje  $0 < \eta < \frac{2}{\beta\sigma_{\max}^2(X)}$  y cualquier punto inicial  $w_0$ . Entonces, se cumplen:*

- 1.  $\lim_{t \rightarrow \infty} \mathcal{L}(w^{(t)}) = 0$ .
- 2.  $\lim_{t \rightarrow \infty} \|w^{(t)}\| = \infty$ .
- 3.  $\forall i \in \{1, \dots, N\} : \lim_{t \rightarrow \infty} y_i w^{(t)T} x_i = \infty$ .

*Demostración.* Dado que el conjunto de entrenamiento  $\mathcal{D}$  es linealmente separable,  $\exists w_* \in \mathbb{R}^D$  de manera que  $\forall i \in \{1, \dots, N\} : y_i w_*^T x_i > 0$ , entonces

$$w_*^T \nabla \mathcal{L}(w) = \sum_{i=1}^N \underbrace{\ell'(y_i w_*^T x_i)}_{<0} \underbrace{y_i w_*^T x_i}_{>0} < 0.$$

Para cualquier  $w$  finito, la suma anterior no puede ser igual a 0, ya que es una suma de términos negativos ( $\forall i : y_i w_*^T x_i > 0$ ) y  $\forall u : \ell'(u) < 0$ , debido a las propiedades que cumple la función de pérdida  $\ell$ . Por tanto, no hay puntos críticos finitos  $w$  para los cuales  $\nabla \mathcal{L}(w) = 0$ . Sin embargo, el descenso de gradiente en una función de pérdida suave con una tasa de aprendizaje adecuada siempre garantiza converger a un punto crítico, es decir,  $\nabla \mathcal{L}(w^{(t)}) \rightarrow 0$  (véase Lema 6.7).

Este hecho implica necesariamente que  $\|w^{(t)}\| \rightarrow \infty$ , que es la condición (2), mientras se verifique que  $\exists t_0 : \forall t > t_0, \forall i : y_i w_t^T x_i > 0$ , que es la condición (3), pues únicamente entonces se cumple  $\ell'(y_i w^{(t)}^T x_i) \rightarrow 0$ . Entonces, se verifica que  $\mathcal{L}(w^{(t)}) \rightarrow 0$  (condición (1)) y, por tanto, el descenso de gradiente converge hacia un mínimo global.  $\square$

El Teorema 6.8 nos asegura que el descenso de gradiente, eligiendo una tasa de aprendizaje adecuada, converge hacia un mínimo global, dado que  $\mathcal{L}(w^{(t)}) \rightarrow 0$ . Además, no se requiere que la función de pérdida  $\mathcal{L}(w)$  sea convexa, lo que amplia la aplicabilidad del resultado a una clase más general de funciones.

### 6.3. Optimización en la zona sobreparametrizada

Como hemos observado en secciones anteriores, la sobreparametrización resulta beneficiosa dentro del marco del aprendizaje estadístico. Este comportamiento, aunque contraintuitivo: “Un modelo con cero error de entrenamiento está por encima de los datos de entrenamiento y, por lo general, generalizará mal” [HTF01], refleja cómo las soluciones de los sistemas sobreparametrizados pueden mejorar la generalización. Sin embargo, desde el punto de vista de la optimización, que se concentra en los algoritmos y técnicas utilizadas para encontrar el mejor modelo posible, la sobreparametrización también ofrece ventajas, ya que facilita la convergencia de los métodos de optimización hacia un mínimo global, especialmente en aquellos que emplean el descenso de gradiente o alguna extensión del mismo.

En el caso de que nuestra función de pérdida  $\ell : \mathcal{Y} \rightarrow \mathbb{R}$  sea convexa y nuestro modelo  $g : \mathcal{X} \rightarrow \mathcal{Y}$ , elegido del conjunto de hipótesis, sea lineal, es fácil observar que  $l \circ f$  es convexa (basta con aplicar las propiedades de la linealidad de  $g$  junto con las propiedades de convexidad de  $\ell$ ). Como resultado, se garantiza la convergencia hacia la mejor solución, es decir, el mínimo global de la función de pérdida, sin riesgo de quedar atrapados en mínimos locales.

No obstante, en el marco del aprendizaje profundo, el modelo resultante es no lineal debido al uso de funciones de activación no lineales, lo que provoca que, en general, **el paisaje de la función de pérdida sea no convexo**. Esto implica que los métodos de optimización de primer orden, como el GD, puedan quedar atrapados en mínimos locales, ya que solo utilizan información local del gradiente, dependiendo así de su inicialización. Como resultado, los sistemas sobreparametrizados dan lugar a paisajes de la función de pérdida que son **esencialmente no convexos**, es decir, por lo general no existe un entorno alrededor de ningún minimizador global en el que el paisaje de la función de pérdida sea convexo. Esto

## 6. Análisis Teórico del Deep Double Descent

contrasta con lo que ocurre en los sistemas infraparametrizados, donde dicho entorno suele existir, aunque sea extremadamente pequeño (véase Figura 6.4).

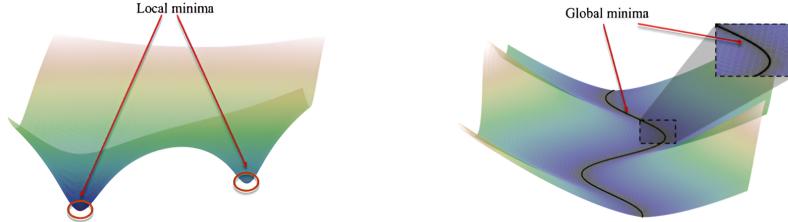


Figura 6.4.: Ejemplos de paisajes de la función de pérdida en redes neuronales [LZB21]. A la izquierda, se muestra un paisaje localmente convexo alrededor de los mínimos locales (*local minima*), ligado a la zona infraparametrizada. A la derecha, se muestra un paisaje incompatible con la convexidad local con múltiples mínimos globales (*global minima*), ligado a la zona sobreparametrizada.

Con el fin de estudiar de manera más formal los resultados presentados, consideramos nuevamente un problema típico del aprendizaje supervisado, con un conjunto de datos  $\mathcal{D}$  compuesto por  $N$ -pares de ejemplos, es decir,  $\mathcal{D} = \{(x_i, y_i)\}_1^N$ , con  $x_i \in \mathbb{R}^D$  e  $y \in \mathbb{R}$ . Si consideramos nuestro conjunto de hipótesis  $\mathcal{H}$  como una familia paramétrica de modelos, como, por ejemplo, una red neuronal, nuestro objetivo es encontrar el vector de parámetros óptimo  $w^*$ , de manera que el modelo se ajuste a los datos de entrenamiento, es decir:

$$f(w^*; x_i) \approx y_i, \quad i \in \{1, \dots, n\}.$$

Matemáticamente, esto es equivalente a resolver un sistema con  $N$  ecuaciones<sup>1</sup>. Agregándolas todas en una única función, podemos escribir:

$$\mathcal{F}(w) = y, \text{ donde } w \in \mathbb{R}^P, y \in \mathbb{R}^N, \mathcal{F}(\cdot) : \mathbb{R}^P \rightarrow \mathbb{R}^N. \quad (6.5)$$

De esta manera,  $\mathcal{F}_i(w) = f(w; x_i)$ . Una solución exacta de la ecuación (6.5) corresponde a la interpolación.

Usamos  $\mathcal{DF}$  para representar la derivada de la función  $\mathcal{F}$ , donde  $\mathcal{DF} \in \mathcal{M}_{N \times P}(\mathbb{R})$ , con  $(\mathcal{DF})_{ij} = \frac{\partial \mathcal{F}_i}{\partial w_j}$ . Denotamos la matriz hessiana de la función  $\mathcal{F}$  como  $H_{\mathcal{F}}$ , que es un tensor de tamaño  $N \times P \times P$  con  $(H_{\mathcal{F}})_{ijk} = \frac{\partial^2 \mathcal{F}_i}{\partial w_j \partial w_k}$ , y definimos la norma del tensor hessiano como  $\|H_{\mathcal{F}}\| = \max_{i \in \{1, \dots, N\}} \|H_{\mathcal{F}_i}\|$ , donde  $H_{\mathcal{F}_i} = \frac{\partial^2 \mathcal{F}_i}{\partial w^2}$ . De manera análoga, denotamos la matriz hessiana de la función de pérdida como  $H_{\mathcal{L}} = \frac{\partial^2 \mathcal{L}}{\partial w^2}$ .

Para detallar de manera analítica la no convexidad en la zona sobreparametrizada, se presenta el siguiente resultado.

**Proposición 6.9.** *Sea  $w^*$  una solución al problema planteado, es decir,  $\mathcal{L}(w^*) = 0$ , y supongamos que  $\frac{d}{dw} \frac{\partial \mathcal{L}}{\partial w}(w^*) \neq 0$  y  $\text{rang}(H_{\mathcal{F}_i}(w^*)) > 2N$  para algún  $i \in \{1, \dots, N\}$ . Entonces  $\mathcal{L}(w)$  no es convexa en ningún entorno de  $w^*$ .*

<sup>1</sup>Para problemas de clasificación multiclas o problemas de regresión con múltiples salidas, donde  $C$  es el número de clases o salidas distintas, estaremos resolviendo un sistema de  $N \times C$  ecuaciones.

*Demostración.* Se presenta una idea intuitiva de cómo realizar la demostración del resultado anterior. Para ello, tomaremos dos puntos distintos del entorno de la solución  $w^*$  y evaluaremos las matrices hessianas de la función de pérdida en dichos puntos. Una vez evaluadas, podemos verificar que ambas matrices no pueden ser no negativas de manera simultánea, lo que implica que la función de pérdida  $\mathcal{L}(w)$  no es localmente convexa alrededor de  $w^*$ , ya que la hessiana no es semidefinida positiva en ese entorno, lo cual es una condición necesaria para que la función sea localmente convexa.

Por último, se referencia al lector más curioso al Apéndice A para la demostración detallada de la proposición.  $\square$

Para el caso de sistemas infraparametrizados, los mínimos locales se encuentran generalmente aislados. Dado que  $H_{\mathcal{L}}(w^*)$  es definida positiva cuando  $w^*$  es un mínimo aislado, por la continuidad de  $H_{\mathcal{L}}(\cdot)$ , la definición de positividad se preserva en el entorno de  $w^*$ . Por consiguiente,  $\mathcal{L}(w)$  es localmente convexa alrededor de  $w^*$ .

A su vez, en los sistemas sobreparametrizados contamos con más parámetros que constantes. En este caso, el sistema de ecuaciones general tiene múltiples soluciones exactas, las cuales forman una variedad continua de dimensión  $P - N > 0$ , de modo que ninguna de estas soluciones está aislada. Este hecho puede observarse en la Figura 6.4, y se encuentra demostrado en el Apéndice A de [LZB21]. En cuanto a nuestro interés, podemos concluir que, para redes suficientemente parametrizadas, **los mínimos globales siempre están rodeados por otro mínimo global en un entorno relativamente pequeño**.

Estos resultados nos conducen a la conclusión de que, para el análisis de los sistemas sobreparametrizados, no podemos utilizar la convexidad, ni siquiera localmente, como base. En consecuencia, es necesario buscar una condición, a priori, más simple, que verifiquen las funciones de pérdida con el fin de analizar el comportamiento de dichos sistemas. Para ello, se recurre a una modificación de la condición de Polyak y Lojasiewicz.

**Definición 6.10** (Condición  $\mu$ -PL). Decimos que una función no negativa  $\mathcal{L}$  verifica la condición  $\mu$ -PL (Polyak-Lojasiewicz) en un conjunto  $\mathcal{S} \subset \mathbb{R}^P$  para  $\mu > 0$ , si

$$\|\nabla \mathcal{L}(w)\|^2 \geq \mu \mathcal{L}(w), \quad \forall w \in \mathcal{S}.$$

**Teorema 6.11.** Supongamos que la función de pérdida  $\mathcal{L}(w)$  es  $\beta$ -suave y cumple la condición  $\mu$ -PL en la bola  $B(w_0, R) = \{w \in \mathbb{R}^P : \|w - w_0\| \leq R\}$ , con  $R = \frac{2\sqrt{2\beta\mathcal{L}(w_0)}}{\mu}$ . Entonces se tiene:

1. (*Existencia de una solución*). Existe una solución (minimizador global de  $\mathcal{L}$ )  $w^* \in B(w_0, R)$ , de manera que  $\mathcal{F}(w^*) = y$ .
2. (*Convergencia del GD*). El gradiente descendente con un learning rate  $\eta \leq \frac{1}{\sup_{w \in B(w_0, R)} \|H_{\mathcal{L}}(w)\|}$  converge a una solución global en  $B(w_0, R)$ :

$$\mathcal{L}(w_t) \leq (1 - \eta\mu)^t \mathcal{L}(w_0).$$

*Demostración.* Probaremos este teorema por inducción, sabiendo que nuestra hipótesis de inducción nos dice que, para todo  $t \geq 0$ ,  $w_t$  está dentro de la bola  $B(w_0, R)$  con  $R = \frac{2\sqrt{2\beta\mathcal{L}(w_0)}}{\mu}$ , y  $\mathcal{L}(w_t) \leq (1 - \eta\mu)^t \mathcal{L}(w_0)$ .

En el caso inicial, cuando  $t = 0$ , es trivial que  $w_0 \in B(w_0, R)$  y que  $\mathcal{L}(w_t) \leq (1 - \eta\mu)^0 \mathcal{L}(w_0)$ . Supongamos que, por inducción, para un  $t \geq 0$ ,  $w_t$  está en la bola  $B(w_0, R)$ . Para probar que  $w_{t+1} \in B(w_0, R)$ , usamos que la función de pérdida  $\mathcal{L}$  es  $\beta$ -suave y, por tanto, verifica:

## 6. Análisis Teórico del Deep Double Descent

$$\mathcal{L}(w_{t+1}) - \mathcal{L}(w_t) - \nabla \mathcal{L}(w_t)(w_{t+1} - w_t) \leq \frac{\beta}{2} \|w_{t+1} - w_t\|^2. \quad (6.6)$$

Tomando  $w_{t+1} - w_t = -\eta \nabla \mathcal{L}(w_t)$  en la ecuación (6.6), nos queda:

$$\mathcal{L}(w_{t+1}) - \mathcal{L}(w_t) + \eta \|\nabla \mathcal{L}(w_t)\|^2 \leq \frac{\beta}{2} \eta^2 \|\nabla \mathcal{L}(w_t)\|^2,$$

lo que implica que

$$\mathcal{L}(w_{t+1}) - \mathcal{L}(w_t) \leq -\left(\eta - \frac{\beta}{2}\eta^2\right) \|\nabla \mathcal{L}(w_t)\|^2.$$

Además, usando el Lema 6.7 para  $\eta \leq \frac{1}{\beta}$ , se verifica

$$\mathcal{L}(w_{t+1}) - \mathcal{L}(w_t) \leq -\frac{\eta}{2} \|\nabla \mathcal{L}(w_t)\|^2 \implies \frac{2(\mathcal{L}(w_t) - \mathcal{L}(w_{t+1}))}{\mu} \geq \|\nabla \mathcal{L}(w_t)\|^2. \quad (6.7)$$

Aplicando la condición  $\mu$ -PL a la primera parte de la ecuación anterior, se satisface

$$-\frac{\eta}{2} \|\nabla \mathcal{L}(w_t)\|^2 \leq -\eta \mu \mathcal{L}(w_t). \quad (6.8)$$

Finalmente, por la hipótesis de inducción, se cumple

$$\mathcal{L}(w_{t+1}) \leq (1 - \eta \mu) \mathcal{L}(w_t) \leq (1 - \eta \mu)^{t+1} \mathcal{L}(w_0).$$

Ahora, podemos analizar la distancia de  $w_{t+1}$  a  $w_0$ :

$$\begin{aligned} \|w_{t+1} - w_0\| &= \eta \left\| \sum_{i=0}^t \nabla \mathcal{L}(w_i) \right\| \leq \eta \sum_{i=0}^t \|\nabla \mathcal{L}(w_i)\| \\ &\leq \eta \sum_{i=0}^t \sqrt{2\beta(\mathcal{L}(w_i) - \mathcal{L}(w_{i+1}))} \quad (\text{usando } \eta = \frac{1}{\beta} \text{ en la ecuación (6.7)}) \\ &\leq \eta \sum_{i=0}^t \sqrt{2\beta \mathcal{L}(w_i)} \\ &\leq \eta \sqrt{2\beta} \left( \sum_{i=0}^t (1 - \eta \mu)^{i/2} \right) \sqrt{\mathcal{L}(w_0)} \quad (\text{usando la ecuación (6.8)}) \\ &\leq \eta \sqrt{2\beta} \sqrt{\mathcal{L}(w_0)} \frac{1}{1 - \sqrt{1 - \eta \mu}} \leq \frac{2\sqrt{2\beta} \sqrt{\mathcal{L}(w_0)}}{\mu} = R. \end{aligned}$$

Por tanto,  $w_{t+1}$  se encuentra en la bola  $B(w_0, R)$ .  $\square$

Finalmente, se presenta una idea intuitiva, utilizando la función de pérdida cuadrática, de por qué los paisajes de pérdida de los sistemas sobreparametrizados satisfacen la condición  $\mu$ -PL en la mayor parte de su espacio de parámetros.

**Definición 6.12.** El núcleo tangente de la función  $\mathcal{F}$  en el vector  $w$  se define como

$$K(w) = \mathcal{D}\mathcal{F}(w) \mathcal{D}\mathcal{F}^T(w),$$

y, entonces,  $K(w) \in \mathcal{M}_{N \times P}(\mathbb{R})$  es una matriz semidefinida positiva.

**Definición 6.13** (Condicionamiento uniforme). Decimos que  $\mathcal{F}(w)$  está  $\mu$ -uniformemente condicionada ( $\mu > 0$ ) en un subconjunto  $\mathcal{S} \subset \mathbb{R}^P$  si el valor propio más pequeño de su núcleo tangente satisface

$$\lambda_{\min}(K(w)) \geq \mu, \quad \forall w \in \mathcal{S}.$$

El siguiente resultado muestra cómo el hecho de estar condicionada de manera uniforme es una condición suficiente para que la pérdida cuadrática cumpla la condición  $\mu$ -PL.

**Teorema 6.14** (Condicionamiento uniforme  $\implies$  Condición  $\mu$ -PL). Si  $\mathcal{F}(w)$  está  $\mu$ -uniformemente condicionada en un conjunto  $\mathcal{S} \subset \mathbb{R}^P$ , entonces la función de pérdida cuadrática  $\mathcal{L}(w) = \frac{1}{2}\|\mathcal{F}(w) - y\|^2$  satisface la condición  $\mu$ -PL\* en  $\mathcal{S}$ .

*Demostración.*

$$\begin{aligned} \frac{1}{2}\|\nabla \mathcal{L}(w)\|^2 &= \frac{1}{2}(\mathcal{F}(w) - y)^T K(w)(\mathcal{F}(w) - y) \\ &\geq \frac{1}{2}\lambda_{\min}(K(w))\|\mathcal{F}(w) - y\|^2 = \lambda_{\min}(K(w))\mathcal{L}(w) \geq \mu\mathcal{L}(w). \end{aligned}$$

□

Nos centramos ahora en mostrar una intuición de por qué la pérdida cuadrática cumple la condición  $\mu$ -PL en la mayor parte del espacio de parámetros. La observación clave viene dada por el rango del núcleo tangente:

$$\text{rang}(K(w)) = \text{rang}((\mathcal{D}\mathcal{F}(w)\mathcal{D}\mathcal{F}^T(w))) = \text{rang}(\mathcal{D}\mathcal{F}(w)),$$

donde  $K(w)$  es, por definición, una matriz semidefinida positiva. Por tanto, el conjunto singular  $\mathcal{S}_{\text{sing}}$ , donde el núcleo tangente es degenerado, se puede escribir como

$$\mathcal{S}_{\text{sing}} = \{w \in \mathbb{R}^P \mid \lambda_{\min}(K(w)) = 0\} = \{w \in \mathbb{R}^P \mid \text{rang}(\mathcal{D}\mathcal{F}(w)) < n\}.$$

Así, tenemos que  $\text{rang}(K(w)) = \min(P, N)$ . Para el caso más simple, consideremos  $N = 1$ . En este caso, el núcleo tangente es un escalar y  $K(w) = \|\mathcal{D}\mathcal{F}(w)\|^2$  es singular si y solo si  $\mathcal{D}\mathcal{F}(w) = 0$ . Como resultado, mediante un análisis de conteo de parámetros, esperamos que el conjunto singular  $\mathcal{S}_{\text{sing}} = \{w \mid K(w) = 0\}$  tenga codimensión  $P$  y, en consecuencia, consista únicamente de puntos aislados.

Aplicando un razonamiento similar, cuando  $P > N$ , esperamos que  $\mathcal{S}_{\text{sing}}$  tenga codimensión positiva ( $P - N + 1$ ) y sea un conjunto de medida cero. Esto significa que, dentro de un conjunto compacto, para valores suficientemente pequeños de  $\mu$ , es probable encontrar puntos que no satisfacen la condición  $\mu$ -PL alrededor de  $\mathcal{S}_{\text{sing}}$ , el cual es un subconjunto de baja dimensión en  $\mathbb{R}^P$ . Esto puede observarse en la Figura 6.5. Es importante notar que, cuanto mayor sea el grado de sobreparametrización del sistema, mayor será la codimensión esperada del conjunto singular. En otras palabras, la región donde la condición  $\mu$ -PL no se cumple se vuelve más pequeña en comparación con el espacio total de parámetros.

En contraste, cuando  $P < N$ , el núcleo tangente es siempre degenerado ( $\lambda_{\min}(K(w)) \equiv 0$ ), por lo que tales sistemas no pueden estar uniformemente condicionados y no pueden satisfacer la condición  $\mu$ -PL.



Figura 6.5.: Ejemplo de función de pérdida que satisface la condición  $\mu\text{-PL}$  [LZB21]. La función de pérdida es  $\mu\text{-PL}$  en la zona sombreada. El conjunto singular corresponde a los vectores  $w$  con núcleo tangente degenerado. Cada bola de radio  $R = O\left(\frac{1}{\mu}\right)$  dentro del dominio  $\mu\text{-PL}$  se interseca con el conjunto de mínimos globales de la función de pérdida.

### 6.3.1. El impacto del sesgo inductivo en la selección de hipótesis

Si bien los resultados presentados anteriormente aseguran la existencia y convergencia, en la zona sobreparametrizada, hacia un minimizador global de la función de pérdida, sabemos que en dicho régimen existe una multitud de minimizadores globales.

A pesar de que todavía no existe una propuesta concluyente para seleccionar, de entre todas las hipótesis de  $S$ , aquella que mejor generaliza, se sabe que, aunque todas sean minimizadoras globales (es decir, satisfacen  $\mathcal{L}(w) = 0$ ), no necesariamente todas garantizan una buena generalización. Un ejemplo claro de esto se puede observar para el caso de regresores polinómicos en la Figura 6.6, donde, una vez que tenemos suficientes parámetros para ajustar todos los datos de entrenamiento, empezamos a obtener distintos modelos que son minimizadores globales.

Extrayendo conclusiones de la Figura 6.6 y apoyándonos en la intuición ampliamente compartida dentro de la comunidad científica, se puede postular que una noción adecuada de **suavidad funcional** podría desempeñar un papel fundamental en la selección del mejor modelo con miras a la generalización.

La idea de maximizar la suavidad de una función sujeta a la interpolación de los datos representa una forma de actuar según el **principio de la navaja de Ockham** [BEHW87]. Este principio establece que se debe preferir la explicación más simple que sea consistente con las evidencias. En nuestro caso, podemos utilizar como analogía que los datos de entrenamiento representan las evidencias, mientras que la simplicidad se asocia con la “suavidad” de la hipótesis a elegir. De esta manera, el principio de la máxima suavidad se puede formular como: “*Elegir la función más suave, de acuerdo con alguna noción de suavidad funcional, entre todas las que son minimizadoras globales*” [Bel21].

Este comportamiento también se encuentra asociado, en cierta medida, al conjunto de hipótesis  $\mathcal{H}$  del modelo. Si consideramos dos conjuntos de hipótesis  $\mathcal{H}_1$  y  $\mathcal{H}_2$ , de manera que  $\mathcal{H}_1 \subset \mathcal{H}_2$ , y sus correspondientes subconjuntos que contienen las hipótesis que son minimizadoras globales,  $S_1 \subset \mathcal{H}_1$  y  $S_2 \subset \mathcal{H}_2$ , entonces, estos conjuntos están relacionados

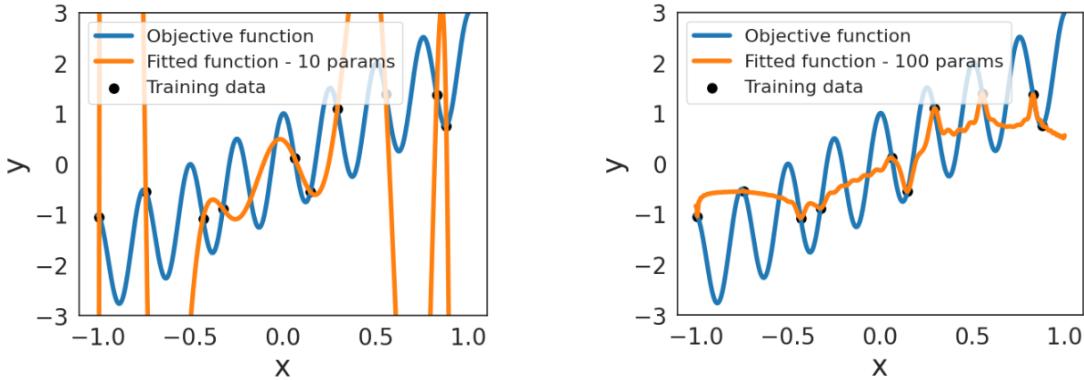


Figura 6.6.: Distintos modelos polinómicos (línea naranja) para aproximar una función objetivo (línea azul), inspirada en [SKR<sup>+</sup>23]. Ambos modelos se ajustan de manera perfecta a los datos de entrenamiento (puntos azules). Sin embargo, el modelo de la derecha generaliza mejor al estar regularizado y suavizado hacia una solución de norma pequeña.

por la misma inclusión ( $S_1 \subset S_2$ ).

Por tanto, si  $\|\cdot\|_s$  es cualquier norma funcional, o más generalmente, cualquier funcional, se verifica que

$$\min_{g \in S_2} \|g\|_s \leq \min_{g \in S_1} \|g\|_s.$$

Suponiendo que  $\|\cdot\|_s$  es el sesgo inductivo correcto, que mide esa suavidad (por ejemplo, una norma de Sobolev), entonces esperamos que la hipótesis seleccionada de  $S_2$  sea más suave, debido a que este conjunto contiene un mayor número de hipótesis que el conjunto  $S_1$  y, por tanto, este conjunto es más adecuado para resolver el problema, lo que lleva a una mejor generalización de la hipótesis elegida.

El principio enunciado anteriormente también permite explicar la aparición del máximo en el error de generalización cerca del umbral de interpolación, ya que en ese punto el modelo se ve forzado a pasar, exactamente, por cada uno de los puntos de entrenamiento. Esto conduce a una solución única que depende exclusivamente de los datos de entrenamiento, lo que obliga al modelo a seleccionar dicha solución independientemente de su “suavidad”.

Ligado a esto, Somepalli et al. [SFB<sup>+</sup>22] analizaron la reproducibilidad de los resultados obtenidos por las mismas redes neuronales a partir del estudio de sus fronteras de decisión. Para nuestro interés, observaron que cerca del umbral de interpolación, y de forma más notable en presencia de ruido, las regiones de decisión tienden a fragmentarse en áreas más pequeñas, las cuales, por lo general, no son reproducibles entre diferentes ejecuciones. En contraste, fuera de este umbral, hay un menor número de regiones de decisión y se encuentran bien definidas, siendo en cierta medida más “suaves” (véase Figura 6.7).

Finalmente, a partir de una puntuación de reproducibilidad basada en dichas regiones de decisión, observaron que esta métrica toma valores más bajos cerca del umbral de interpolación. Esto sugiere que, en dicha región, el modelo seleccionado tiende a ser más particular y depende fuertemente de la inicialización de los parámetros de la red utilizada.

Generalmente, puede pensarse que los sesgos inductivos introducidos por los algoritmos de optimización son de naturaleza restrictiva, es decir, imponen restricciones al espacio de

## 6. Análisis Teórico del Deep Double Descent

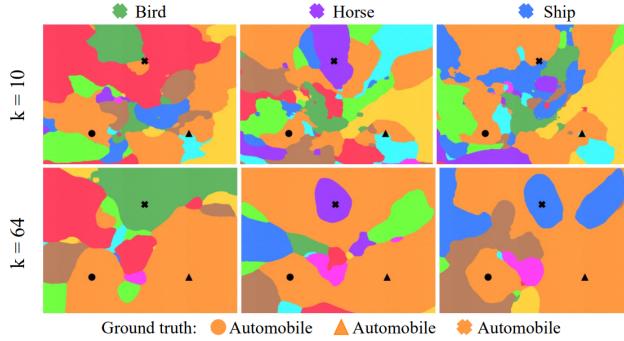


Figura 6.7.: Fronteras de decisión para modelos con distinta capacidad [SFB<sup>+</sup>22]. Cada columna representa una tripleta distinta de imágenes, en la que una imagen de automóvil (marcada con una  $\times$ ) ha sido etiquetada incorrectamente. El modelo infraparametrizado ( $k = 10$ ) genera numerosas regiones de decisión pequeñas y fragmentadas, mientras que el modelo sobreparametrizado ( $k = 64$ ) produce un menor número de regiones, más amplias y claramente definidas.

hipótesis que están alineadas con el problema de interés. En otras palabras, dado que existen múltiples configuraciones del modelo que se ajustan adecuadamente a los datos pero generan una mala generalización, se prefiere restringir el espacio de hipótesis a aquellas configuraciones que tienden a generalizar mejor para el problema en cuestión, independientemente de si estas son óptimas en un sentido más general. Este enfoque está estrechamente relacionado con los principios de regularización.

Al reducir el espacio de hipótesis, este se ve más rápidamente limitado por los datos, ya que existen menos opciones para que el modelo se ajuste a ellos. No obstante, estos sesgos restrictivos pueden considerarse, en ciertos contextos, como indeseables. En efecto, lo que se busca es respaldar cualquier solución capaz de describir adecuadamente los datos, lo que implica favorecer un espacio de hipótesis lo suficientemente flexible.

La idea general de preferir ciertas soluciones sobre otras, incluso cuando ambas se ajustan igualmente a los datos, se conoce como **sesgo inductivo suave** [Wil25]. La Figura 6.8 ilustra este concepto, distinguiéndolo de otros tipos de sesgo. El hecho de considerar estos sesgos inductivos suaves se relaciona directamente con la disminución de la varianza (y, por tanto, también del error) a medida que aumentamos la capacidad del modelo. Este tipo de sesgos se centra en una zona específica (y favorable) del espacio de hipótesis, reduciendo así la varianza asociada. Es comparable a utilizar un sesgo restrictivo, pero enfocado en un subconjunto adecuado de hipótesis, como puede observarse en la Figura 6.8.

Por otro lado, a medida que aumenta la cantidad de parámetros, también crece el volumen de soluciones planas en el paisaje de la función de pérdida (como se ha observado en la subsección anterior), lo que facilita el acceso y la selección de una de ellas [HEG<sup>+</sup>20]. Así, los modelos más grandes (sobreparametrizados) presentan sesgos inductivos más pronunciados, como se puede observar en la Figura 6.9.

Ligado a la explicación del fenómeno de la generalización, también surge el siguiente resultado [MBW20], ampliamente relacionado con lo presentado en la Subsección 6.1.1.

**Definición 6.15.** (Dimensionalidad efectiva) Sea  $A \in \mathcal{M}_{n \times n}(\mathbb{R})$  una matriz simétrica. Se define su dimensionalidad efectiva como sigue:

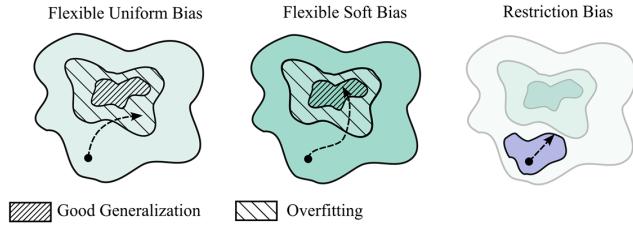


Figura 6.8.: Diferentes tipos de sesgo para el espacio de hipótesis [Wil25]. A la izquierda, observamos un conjunto de hipótesis grande sobre el cual no se tiene preferencia por las soluciones que ajustan igualmente bien los datos. En este caso, el entrenamiento puede derivar en soluciones de sobreajuste que generalizan mal. En el centro, se presenta un conjunto de hipótesis flexible en combinación con un sesgo inductivo suave, que guía el entrenamiento hacia soluciones que generalizan bien. A la derecha, restringir el espacio de hipótesis puede ayudar a evitar el sobreajuste al considerar solo algunas soluciones. Sin embargo, al limitar la expresividad del modelo, se impide captar todos los matices de la realidad, lo que dificulta alcanzar la mejor generalización.

$$N_{eff}(A) = \sum_{i=1}^n \frac{\lambda_i}{\lambda_i + \alpha},$$

donde  $\lambda_i$  son los valores propios de la matriz  $A$  y  $\alpha > 0$  es una constante de regularización.

Así, la dimensionalidad efectiva mide el número de valores propios relativamente grandes, lo que puede interpretarse como una medida de las direcciones significativas en el espacio de parámetros. En particular, la dimensionalidad efectiva de la matriz Hessiana de la función de pérdida, evaluada en el vector de parámetros  $w$ , cuantifica el número de direcciones agudas (direcciones con curvatura) en el paisaje de la función de pérdida, es decir, el número de parámetros que están determinados por los datos.

Maddox et al. [MBW20] descubrieron que, después del entrenamiento, los modelos grandes tienen menos parámetros efectivos que los modelos más pequeños, al observar su dimensionalidad efectiva. En trabajos más recientes, Goldblum et al. (2024) [GFRW24] también demostraron que los LLM presentan un fuerte sesgo hacia soluciones más simples, y que este sesgo es una característica clave para un buen rendimiento. Estos resultados refuerzan la conjectura de que las soluciones más simples tienden a obtener mejores resultados, ya que son más comprensibles y siguen la filosofía de Ockham. Asimismo, el sesgo inductivo hacia estas soluciones contrarresta, de alguna manera, el crecimiento del número de soluciones que no son minimizadoras globales [MRVPL23].

Como conclusión, podemos decir que el uso de sistemas sobreparametrizados resulta ventajoso para obtener cada vez más soluciones posibles (minimizadores globales) para nuestro problema. De esta forma, al aumentar la capacidad del modelo (en nuestro caso, su complejidad efectiva), resultará más sencillo para el algoritmo de optimización elegir una “buena” hipótesis (que generalice mejor que el resto) basada en algún sesgo inductivo que utilice. Este sesgo parece ir ligado hacia soluciones más simples, donde, en el caso de las hipótesis, se pueden entender como más “suaves”, lo que conduce a la hipótesis elegida a una especie de autoregulación.

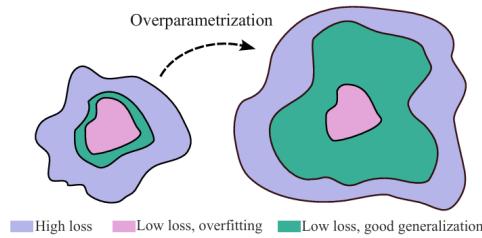


Figura 6.9.: Aumentar la capacidad de un modelo mejora su generalización [Wil25]. Al aumentar el número de parámetros de un modelo, las soluciones “planas” ocupan un mayor volumen dentro del espacio total de hipótesis, lo que implica un sesgo inductivo suave hacia ellas. Aunque los modelos sobreparametrizados presentan muchas hipótesis que sobreajustan los datos, también pueden contener muchas otras que se ajustan bien a los datos y proporcionan una buena generalización. De este modo, la sobreparametrización puede aumentar simultáneamente el tamaño del espacio de hipótesis y del sesgo hacia soluciones más simples.

### 6.3.2. PAC-Bayes y límite de hipótesis numerables

Los marcos de generalización clásicos, como la complejidad de Rademacher o la dimensión VC (véase Sección 4.5), desafían, por tanto, las nuevas nociones y límites de generalización, ya que penalizan el tamaño total del conjunto de hipótesis. Esto sugiere una recomendación hacia sesgos restrictivos, en lugar de promover sesgos inductivos suaves, como se mencionó anteriormente. Por esta razón, en esta subsección buscamos introducir nuevos marcos que se alineen con las ideas más recientes, presentados en [Wil25].

Estos nuevos marcos de generalización, aunque no son novedosos y han existido durante décadas, incluyen enfoques como PAC-Bayes [McA99] y límites de hipótesis numerables [Val84]. Estos ofrecen un enfoque convincente para modelos sobreparametrizados, ya que se centran en identificar qué hipótesis son probables, en lugar de penalizar el tamaño del espacio de hipótesis. De este modo, siguen la tendencia de los sesgos inductivos suaves.

**Teorema 6.16** (Límite de hipótesis numerables). *Sea el riesgo acotado  $\mathcal{L}(g) \in [a, a + \Delta]$  y un espacio de hipótesis numerable  $\mathcal{H}$  para el cual se dispone de una distribución a priori  $P(g)$  de la hipótesis  $g$ . Sea  $\mathcal{L}_{emp}(g)$  el riesgo empírico y  $\mathcal{L}(g) = \mathbb{E}[\mathcal{L}_{emp}(g)]$  el riesgo real. Entonces, con probabilidad al menos  $1 - \delta$ ,*

$$\mathcal{L}(g) \leq \mathcal{L}_{emp}(g) + \Delta \sqrt{\frac{\log\left(\frac{1}{P(h)}\right) + \log\left(\frac{1}{\delta}\right)}{2n}}. \quad (6.9)$$

Esta cota está relacionada con la cota para espacios finitos de hipótesis, pero incluye una distribución a priori  $P(h)$  que pondera las distintas hipótesis, y un espacio de hipótesis numerable en lugar de finito. Además, podemos usar cualquier distribución a priori para evaluar la cota, sin necesidad de que contenga la hipótesis objetivo.

A partir de la ecuación (6.9), podemos derivar cotas más informativas utilizando una distribución a priori de Solomonoff  $P(g) \leq 2^{-K(g|A)}$  [Sol64], donde  $K$  es la **complejidad de Kolmogorov libre de prefijo** de  $g$  (longitud del programa más corto que produce  $g$ , utilizando un lenguaje de programación fijo) al tomar como entrada la arquitectura del

modelo *A*. Sustituyendo esta cota en la ecuación (6.9), obtenemos:

$$\mathcal{L}(g) \leq \mathcal{L}_{emp}(g) + \Delta \sqrt{\frac{K(g|A) \log(2) + \log\left(\frac{1}{\delta}\right)}{2n}}. \quad (6.10)$$

Aunque en general no podemos calcular el programa más corto, sí podemos acotar la arquitectura y cualquier constante no determinada por los datos dentro de la distribución a priori, utilizando la probabilidad condicionada  $K(g|A)$ . Esta complejidad puede ser convertida desde la versión libre de prefijo a la complejidad de Kolmogorov estándar, obteniendo la siguiente cota superior:

$$\log\left(\frac{1}{P(g)}\right) \leq K(g|A) \log(2) \leq C(g) \log(2) + 2 \log(C(g)),$$

donde  $C(g)$  es el número de bits requeridos para representar la hipótesis  $g$  usando algún código específico predefinido. Por tanto, incluso modelos con mucha capacidad pueden tener garantías sólidas de generalización a través de la ecuación (6.10).

Las cotas PAC-Bayes permiten reducir la cantidad de bits desde  $\log_2\left(\frac{1}{P(g)}\right)$  hasta la divergencia de Kullback-Leibler  $KL(Q||P)$ <sup>2</sup>, al considerar una distribución de soluciones deseables  $Q$ . Estos marcos de generalización han sido adaptados para ofrecer garantías de generalización no triviales en modelos con millones de parámetros, incluyendo redes neuronales y grandes modelos de lenguaje (LLMs) [LFK<sup>+</sup>24b].

En resumen, estos marcos sugieren que las cotas de generalización pueden interpretarse de la siguiente forma:

$$\text{Riesgo real} \leq \text{Riesgo empírico} + \text{Compresibilidad del modelo},$$

donde la compresibilidad proporciona una cierta formalización de complejidad.

Este enfoque se relaciona con la **inducción de Solomonoff**, que propone un procedimiento de aprendizaje ideal y sobreparametrizado al máximo, sin límite en la complejidad de la hipótesis [Sol64]. Al asignar pesos exponencialmente mayores a programas más simples, la inducción de Solomonoff garantiza que, aunque el espacio de hipótesis sea enorme, la hipótesis elegida será simple si se ajusta bien a los datos.

En conclusión, estas “nuevas” cotas de generalización tienden a volverse más estrictas, y no más laxas como ocurre con las cotas clásicas, al utilizar modelos más grandes. Esto se debe a que dichos modelos tienden a seleccionar soluciones más simples, en línea con la noción de sesgos inductivos suaves, como se discutió en secciones anteriores. Además, estos límites pueden ser sorprendentemente precisos [LFK<sup>+</sup>24a].

## 6.4. Aproximación no lineal

En esta sección, introduciremos algunos elementos fundamentales de la teoría de la aproximación no lineal [DeV98], la cual está relacionada con el problema del aprendizaje profundo. Además, desarrollaremos ciertos conceptos con el propósito de demostrar la existencia de analogías entre esta teoría y el *Deep Double Descent*, así como de proporcionar una posible explicación desde esta perspectiva.

<sup>2</sup>La divergencia de Kullback-Leibler ( $KL(Q||P)$ ) es una medida no simétrica del grado de diferencia o similitud entre dos distribuciones de probabilidad ( $Q$  y  $P$ ).

## 6. Análisis Teórico del Deep Double Descent

Comenzamos recordando que el problema principal de la teoría de la aproximación es encontrar una representación para una función, generalmente compleja, denominada función objetivo  $f$ , utilizando funciones más simples llamadas **aproximantes**. De esta manera, al igual que en el caso del aprendizaje, ambos enfoques buscan aproximar una determinada función a partir de otras más sencillas.

La aproximación no lineal implica que los aproximantes no provienen de espacios lineales, sino de variedades no lineales. Dado que el incremento en la resolución de la función objetivo solo puede lograrse aumentando la complejidad de los aproximantes, resulta lógico considerar que la aproximación no lineal ofrece ventajas sobre los métodos clásicos. Para evaluar esto, estudiaremos la tasa de aproximación, definida como la disminución del error en función del número de parámetros en el aproximante, en relación con el doble descenso.

Por tanto, la idea de no restringir las aproximaciones a espacios lineales consiste en utilizar una malla más fina en las regiones donde la función objetivo no es muy suave (presenta singularidades) y una malla más gruesa donde sí lo es. No obstante, será necesario caracterizar formalmente esta noción de suavidad para obtener resultados concluyentes.

Finalmente, nos centraremos en la aproximación no lineal conocida como aproximación de  $n$  términos, que consiste en mantener el control únicamente sobre el número de términos a utilizar para realizar la aproximación, dejando que la selección de los términos dependa de la función objetivo. Esto conduce a un problema de doble etapa, en el que la función objetivo se utiliza tanto para seleccionar una base adecuada de una biblioteca de bases como para elegir la mejor aproximación de  $n$  términos en relación con dicha base.

### 6.4.1. Aproximación en un espacio de Hilbert

Dado que los problemas de la teoría de la aproximación son más sencillos de analizar en un espacio de Hilbert, comenzaremos con una breve discusión sobre este entorno, con el fin de introducir los conceptos clave.

Por ello, sea  $\mathcal{H}$  un espacio de Hilbert con producto interno  $\langle \cdot, \cdot \rangle$  y norma  $\|\cdot\|_{\mathcal{H}}$ , y sea  $\{\eta_k\}_{k=1}^n$  una base ortonormal de  $\mathcal{H}$ . De esta manera, en el caso en el que el espacio  $\mathcal{H}$  tenga dimensión infinita, la base ortonormal también será infinita. Para el caso de la aproximación lineal, consideramos el espacio lineal  $\mathcal{H}_n = \text{span}\{\eta_k : 1 \leq k \leq n\}$  para aproximar un elemento  $f \in \mathcal{H}$ . El error de aproximación se mide mediante

$$E_n(f)_{\mathcal{H}} = \inf_{g \in \mathcal{H}_n} \|f - g\|_{\mathcal{H}}.$$

Como contraparte en la aproximación no lineal, que es el caso de estudio de este análisis, contamos con la aproximación de  $n$  términos, que reemplaza  $\mathcal{H}_n$  por el espacio  $\Sigma_n$ , el cual consiste en todos los elementos  $g \in \mathcal{H}$  que pueden expresarse como

$$g = \sum_{k \in \Lambda} c_k \eta_k,$$

donde  $\Lambda \subset \mathbb{N}$  es un conjunto de índices con  $\#\Lambda \leq n$  (cardinalidad finita).

Es claro que, a diferencia de  $\mathcal{H}_n$ , el espacio  $\Sigma_n$  no es lineal. De manera análoga al error de aproximación lineal  $E_n$ , definimos el error de aproximación de  $n$  términos como

$$\sigma_n(f)_{\mathcal{H}} = \inf_{g \in \Sigma_n} \|f - g\|_{\mathcal{H}}.$$

**Definición 6.17.** Dado un número real  $\alpha > 0$ , denotamos por  $\mathcal{A}^\alpha(\Sigma_n)$  la clase de funciones  $f \in \mathcal{H}$  que cumplen la siguiente desigualdad para algún  $M > 0$ :

$$\sigma_n(f)_{\mathcal{H}} \leq Mn^{-\alpha}, \quad n \in \mathbb{N}.$$

Además, definimos la seminorma  $|f|_{\mathcal{A}^\alpha(\Sigma_n)}$  como el ínfimo de todos los valores de  $M$  para los cuales se cumple la ecuación anterior.

Denominaremos al conjunto  $\mathcal{A}^\alpha$  como un **espacio de aproximación**, el cual agrupa las funciones  $f \in \mathcal{H}$  con un comportamiento de aproximación similar, es decir, aquellas cuyo error de aproximación disminuye con una tasa de orden  $n^{-\alpha}$ .

Finalmente, podemos caracterizar el espacio  $\mathcal{A}^\alpha(\Sigma_n)$  como una serie en términos de la base ortonormal de  $\mathcal{H}$ , reorganizando dicha base para que los coeficientes de la función  $f$  sean decrecientes. Por tanto, denotamos  $f = \sum_{k=1}^{\infty} \gamma_k(f) \eta_k$  una vez reordenada la base de modo decreciente en los coeficientes  $\gamma_k(f)$ . De esta manera, representamos por  $\gamma_k(f)$  al valor absoluto del  $k$ -ésimo coeficiente de la función  $f$ .

**Teorema 6.18.** Una función  $f \in \mathcal{A}^\alpha(\Sigma_n)$  si y solo si se cumple la siguiente desigualdad:

$$\gamma_n(f) \leq Mn^{-\alpha-1/2}, \quad (6.11)$$

donde el ínfimo de todos los  $M$  que satisfacen la ecuación anterior es equivalente a la seminorma  $|f|_{\mathcal{A}^\alpha(\Sigma_n)}$ .

*Demostración.* En primer lugar, notemos que

$$\sigma_n(f)_{\mathcal{H}}^2 = \sum_{k>n} \gamma_k(f)^2,$$

dado que el error de aproximación al cuadrado es la suma de los cuadrados de los coeficientes que no están involucrados en la aproximación. Por tanto, si  $f$  cumple la hipótesis (6.11), entonces

$$\sum_{k>n} \gamma_k(f)^2 \leq M^2 \sum_{k>n} k^{-2\alpha-1} = M^2 \sum_{k>n} \frac{1}{k^{2\alpha+1}}. \quad (6.12)$$

Es decir,  $\sum_{k>n} \frac{1}{k^{2\alpha+1}}$  es una serie armónica convergente, ya que  $\alpha > 0$ . De esta manera, se puede acotar la ecuación (6.12) por

$$M^2 \sum_{k>n} \frac{1}{k^{2\alpha+1}} \leq M^2 \int_n^{\infty} \frac{1}{x^{2\alpha+1}} dx = \frac{M^2}{2\alpha} n^{-2\alpha}.$$

En consecuencia,

$$\sigma_n(f)_{\mathcal{H}}^2 \leq \frac{M^2}{2\alpha} n^{-2\alpha} \implies \sigma_n(f)_{\mathcal{H}} \leq \frac{M}{\sqrt{2\alpha}} n^{-\alpha} = CMn^{-\alpha},$$

con  $C = \frac{1}{\sqrt{2\alpha}}$ , lo que demuestra que  $f \in \mathcal{A}^\alpha(\Sigma_n)$ . Por otro lado, si  $f \in \mathcal{A}^\alpha(\Sigma_n)$ , entonces se satisface la siguiente desigualdad:

$$\gamma_{2n}(f)^2 \leq n^{-1} \sum_{m=n+1}^{2n} \gamma_m(f)^2 \leq n^{-1} \sigma_n(f)_{\mathcal{H}}^2 \leq |f|_{\mathcal{A}^\alpha(\Sigma_n)}^2 n^{-2\alpha-1} \leq M^2 n^{-2\alpha-1}.$$

## 6. Análisis Teórico del Deep Double Descent

Una desigualdad análoga se verifica para los coeficientes impares  $\gamma_{2n+1}(f)$ , debido a la naturaleza decreciente de los mismos. De esta forma, la suma de los cuadrados de los coeficientes (ya sean pares o impares) está acotada de manera similar. Finalmente, aplicando raíces a los resultados obtenidos se obtiene la implicación restante.  $\square$

A su vez, definimos el espacio de aproximación  $\mathcal{A}_\infty^\alpha(\Sigma_n)$  como el conjunto de todas las funciones  $f \in \mathcal{H}$  tales que

$$\sup_{n \geq 1} n^\alpha \sigma_n(f)_\mathcal{H} < \infty.$$

Es decir, este conjunto está compuesto por todas las funciones  $f \in \mathcal{H}$  que cumplen que

$$\sigma_n(f)_\mathcal{H} = O(n^{-\alpha}), \quad n \rightarrow \infty.$$

Esto significa que, para las funciones  $f$  en  $\mathcal{A}_\infty^\alpha(\Sigma_n)$ , la sucesión de los errores de aproximación  $\sigma_n(f)_\mathcal{H}$  decrece de manera suficientemente rápida conforme  $n$  crece.

Al mismo tiempo, denotamos por  $\ell_{p,q}$  los espacios de sucesiones de Lorentz<sup>3</sup>. Para nuestro desarrollo, nos centraremos únicamente en el espacio  $\ell_{p,\infty}$  (el espacio débil  $\ell_p$ ), que está formado por todas las sucesiones que satisfacen

$$x^*(n) \leq Mn^{-1/p},$$

donde  $x^*(n)$  es el reordenamiento decreciente de la sucesión  $|x(n)|$ .

Finalmente, el siguiente resultado nos ayuda a relacionar el espacio de aproximación  $\mathcal{A}_\infty^\alpha$  con el espacio débil  $\ell_p$  ( $\ell_{p,\infty}$ ).

**Teorema 6.19.** *Para una aproximación de  $n$  términos en un espacio de Hilbert  $\mathcal{H}$ , una función  $f \in \mathcal{A}_\infty^\alpha$  si y solo si sus coeficientes  $\{f_k : k \in \mathbb{N}\}$  están en el espacio débil  $\ell_\tau$  (es decir, en  $\ell_{\tau,\infty}$ ) con  $\tau = (\alpha + 1/2)^{-1}$ . Además, en este caso, se verifica*

$$C_1 |f|_{\mathcal{A}_\infty^\alpha(\mathcal{H})} \leq \| (f_k) \|_{\ell_{\tau,\infty}} \leq C_2 |f|_{\mathcal{A}_\infty^\alpha(\mathcal{H})},$$

donde  $C_1, C_2 > 0$  son constantes.

### 6.4.2. Aproximación altamente no lineal

Nos preguntamos ahora cómo depende la efectividad de la aproximación de  $n$  términos de la base elegida y si se puede conseguir alguna ventaja eligiendo de manera progresiva una base que dependa de la función objetivo. Para ello, llamamos **biblioteca** a una clase  $\mathcal{L}$  de bases. Por simplicidad, centraremos nuestra discusión en aproximaciones en un espacio de Hilbert  $\mathcal{H}$  y en bibliotecas de bases ortonormales para  $\mathcal{H}$ .

Por consiguiente, nuestro problema de aproximación consiste, dada una función objetivo  $f \in \mathcal{H}$ , en elegir una base  $B \in \mathcal{L}$  y una aproximación de  $n$  términos a  $f$  usando esta base. Llamamos a este tipo de problema de aproximación altamente no lineal, ya que implica una capa adicional de no linealidad en la selección de bases.

No obstante, otra forma de aproximación estrechamente relacionada es la aproximación utilizando  $n$  términos desde un **diccionario**  $\mathbb{D} \subset \mathcal{H}$  de funciones, que será nuestro principal

---

<sup>3</sup>Los espacios de Lorentz son generalizaciones de los espacios  $L^p$ , en el sentido de que  $\ell_{p,p} = L^p$ , y permiten una caracterización más precisa del comportamiento asintótico de las sucesiones. En particular, los espacios  $\ell_{p,\infty}$  corresponden a la versión débil de los espacios  $L^p$ .

objeto de estudio. En nuestro caso, un diccionario se define como un subconjunto arbitrario de  $\mathcal{H}$ .

Por ejemplo, en el caso de redes neuronales, un diccionario de funciones podría tener la siguiente forma:

$$\{\sigma(w \cdot x + b) : w \in \mathbb{R}^P, b \in \mathbb{R}\},$$

donde  $\sigma$  es una función de activación de una variable fija,  $w$  es el vector de parámetros y  $b$  es el término de sesgo. De esta manera, se aproximan funciones de  $P$  variables mediante combinaciones lineales de funciones de dicho conjunto. Generalmente, se requiere que  $\sigma$  cumpla con algunas propiedades adicionales (véase Subsección 3.2.1.4).

Otro ejemplo particular de diccionario es utilizar una base ortonormal  $B$  para  $\mathcal{H}$  y definir el diccionario asociado como sigue:

$$\mathbb{D} = \{\pm b : b \in B\},$$

donde diremos que  $\mathbb{D}$  es el diccionario generado por la base  $B$ .

La característica común de todos los diccionarios es que la familia de funciones utilizada para el proceso de aproximación es **redundante**. Es decir, hay muchas más funciones en el diccionario de las necesarias para aproximar cualquier función objetivo  $f \in \mathcal{H}$ . Por tanto, buscamos que esta redundancia incremente la eficiencia de la aproximación, aunque, por otro lado, también podría ralentizar el proceso de búsqueda de buenas aproximaciones.

#### 6.4.2.1. Selección óptima de bases

En esta subsección trataremos cómo la selección adecuada de una base influye directamente en el error de aproximación. Asimismo, observaremos que, al disponer de una biblioteca de bases, el error de aproximación se minimiza para cualquier aproximación realizada, ya que se consideran todas las bases disponibles.

Sea  $B = \{\eta_k\}$  una base ortonormal para  $\mathcal{H}$  y sea  $\Sigma_n(B)$  el conjunto de funciones en  $\mathcal{H}$  que pueden escribirse como una combinación lineal de  $n$  de las funciones  $\eta_k$ , con  $k \in \mathbb{N}$ . Además, definimos

$$\sigma_n(f, B) = \sigma_n(f, B)_\mathcal{H} = \inf_{S \in \Sigma_n(B)} \|f - S\|_\mathcal{H}$$

como el error de aproximación correspondiente respecto a la base  $B$ .

Recordemos que la disminución de los errores de aproximación  $\sigma_n(f, B)$  está completamente determinada por los coeficientes reorganizados de la función  $f$ . Como antes, sea  $\gamma_k(f, B)$  el  $k$ -ésimo mayor valor absoluto de estos coeficientes. Aplicando el Teorema 6.19 a los coeficientes  $\gamma_n(f, B)$ , sabemos que  $f \in \mathcal{H}$  está en  $\mathcal{A}_\infty^\alpha(\mathcal{H}, B)$ , si y solo si  $(\gamma_n(f, B))$  está en  $\ell_\tau$  con  $\tau = (\alpha + 1/2)^{-1}$ . Además,

$$C_1 |f|_{\mathcal{A}_\infty^\alpha} \leq \|\gamma_n(f, B)\|_{\ell_\tau, \infty} \leq C_2 |f|_{\mathcal{A}_\infty^\alpha}, \quad (6.13)$$

con constantes de equivalencia independientes de la base  $B$  elegida.

Supongamos ahora que  $\mathcal{L} = \{B_1, \dots, B_n\}$  es una biblioteca de bases ortonormales. Definimos el error de aproximación con respecto a  $\mathcal{L}$  como sigue

$$\sigma_n^\mathcal{L}(f)_\mathcal{H} = \inf_{B \in \mathcal{L}} \sigma_n(f, B)_\mathcal{H}.$$

## 6. Análisis Teórico del Deep Double Descent

A la vista de la ecuación (6.13), obtenemos la siguiente cota superior

$$\sigma_n^{\mathcal{L}}(f)_{\mathcal{H}} \leq Cn^{-\alpha} \inf_B \|\gamma_n(f, B)\|_{\ell_r, \infty}, \quad (6.14)$$

con  $C$  una constante absoluta. Además, para cualquier  $\alpha > 0$ , se tiene

$$\bigcap_B \mathcal{A}_\infty^\alpha(\mathcal{H}, B) \subset \mathcal{A}_\infty^\alpha(\mathcal{H}, \mathcal{L}). \quad (6.15)$$

En consecuencia, para cada base ortonormal  $B$ , la condición  $(\gamma_n(f)) \in \ell_{\tau, \infty}$ ,  $\tau = (\alpha + 1/2)^{-1}$ , puede verse como una condición de suavidad de la función  $f$  relativa a la base  $B$ . Así, el ínfimo en el lado derecho de la ecuación (6.14) puede interpretarse como el ínfimo de condiciones de suavidad relativas a las diferentes bases  $B$ .

De manera similar, podemos ver las clases  $\mathcal{A}_\infty^\alpha(\mathcal{H}, B)$  como clases de suavidad con respecto a la base  $B$ . De esta forma, una ventaja de la selección óptima de bases es que siempre podemos elegir la base  $B \in \mathcal{L}$  en la que  $f$  sea más suave.

Por su parte, las ecuaciones (6.14) y (6.15) muestran que la base más eficiente para aproximar  $f$  depende del número de términos utilizados para la aproximación. De esta forma, se pueden construir dos bases,  $B_1$  y  $B_2$ , y una función objetivo  $f$  de tal manera que, para ciertos valores de  $n$ , la elección óptima de base varíe entre  $B_1$  y  $B_2$ . Es decir, para algunos valores de  $n$ ,  $B_1$  ofrecerá un mejor error de aproximación, mientras que para otros,  $B_2$  es más eficiente, lo que demuestra que no hay una base universalmente mejor para todos los valores de  $n$ .

### 6.4.2.2. Aproximación usando $n$ términos de un diccionario

Supongamos que  $\mathbb{D}$  es un diccionario de funciones en  $\mathcal{H}$ . Para simplificar, asumimos (sin pérdida de generalidad en la aproximación de  $n$  términos) que cada  $g \in \mathbb{D}$  tiene norma uno, es decir,  $\|g\|_{\mathcal{H}} = 1$ , y que  $-g \in \mathbb{D}$  siempre que  $g \in \mathbb{D}$ .

Para cada  $n \in \mathbb{N}$ , definimos  $\Sigma_n = \Sigma_n(\mathbb{D})$  como la colección de todas las funciones en  $\mathcal{H}$  que pueden expresarse como una combinación lineal de, como máximo,  $n$  elementos de  $\mathbb{D}$ . Así, cada función  $S \in \Sigma_n$  puede escribirse de la forma:

$$S = \sum_{g \in \Lambda} c_g g, \quad \Lambda \subset \mathbb{D}, \quad \#\Lambda \leq n, \quad (6.16)$$

donde  $c_g \in \mathbb{R}$  son los coeficientes. Cabe destacar que un elemento de  $\Sigma_n(\mathbb{D})$  puede expresarse según la expresión (6.16) de más de una forma.

**Definición 6.20.** Para una función  $f \in \mathcal{H}$ , definimos su error de aproximación a partir de un diccionario  $\mathbb{D}$  como:

$$\sigma_n(f) = \sigma_n(f, \mathbb{D})_{\mathcal{H}} = \inf_{S \in \Sigma_n} \|f - S\|_{\mathcal{H}}.$$

Así, nuestro interés se centra en obtener estimaciones para  $\sigma_n$  (tanto superiores como inferiores). Para este propósito, introducimos la siguiente forma de medir la suavidad con respecto a un diccionario  $\mathbb{D}$ .

**Definición 6.21.** Para un diccionario  $\mathbb{D}$  y cualquier  $\tau > 0$ , definimos la clase de funciones:

$$\mathcal{K}_\tau^o(\mathbb{D}, M) = \left\{ f = \sum_{g \in \Lambda} c_g g : \Lambda \subset \mathbb{D}, \quad \#\Lambda < \infty \quad \text{y} \quad \sum_{g \in \Lambda} |c_g|^\tau \leq M^\tau \right\},$$

y definimos  $\mathcal{K}_\tau(\mathbb{D}, M)$  como la clausura en  $\mathcal{H}$  de  $\mathcal{K}_\tau^o(\mathbb{D}, M)$ . Además, definimos  $\mathcal{K}_\tau(\mathbb{D})$  como la unión de las clases  $\mathcal{K}_\tau(\mathbb{D}, M)$  para todo  $M > 0$ . Para  $f \in \mathcal{K}_\tau(\mathbb{D})$ , definimos la seminorma:

$$|f|_{\mathcal{K}_\tau(\mathbb{D})} = \inf\{M : f \in \mathcal{K}_\tau(\mathbb{D}, M)\}.$$

En otras palabras,  $\mathcal{K}_\tau^o(\mathbb{D}, M)$  está formado por la clase de funciones que se pueden escribir como una combinación lineal de un número finito de elementos de  $\mathbb{D}$ , de tal manera que la suma de los coeficientes elevados a la potencia  $\tau$  se encuentre acotada, controlando así la magnitud de las combinaciones lineales.

Por otro lado,  $|f|_{\mathcal{K}_\tau(\mathbb{D})}$  nos indica la dificultad de aproximar una función  $f$  utilizando una combinación de funciones del diccionario. Cuanto más pequeña sea la seminorma, más sencillo será aproximar la función  $f$ .

*Observación 6.22.* Cuando  $\tau = 1$ ,  $\mathcal{K}_1$  es la clase de funciones que son una combinación convexa de las funciones en  $\mathbb{D}$ .

Para el caso en el que  $\mathbb{D}$  es generado por una base  $B$  se sigue que la aproximación de  $n$  términos desde  $\mathbb{D}$  es la misma que la aproximación de  $n$  términos desde  $B$ . En particular, si  $\tau = (\alpha + 1/2)^{-1}$  y  $B$  es ortonormal, se verifica que:

$$\sigma_n(f, \mathbb{D})_{\mathcal{H}} \leq Cn^{-\alpha}|f|_{\mathcal{K}_\tau(\mathbb{D})}. \quad (6.17)$$

Sabemos que  $Cn^{-\alpha}$  es un factor de decaimiento que describe cómo el error de aproximación disminuye a medida que incrementamos el número de términos  $n$  en la aproximación. La constante  $C$  es un valor positivo que depende tanto del diccionario como de la función, mientras que la potencia  $\alpha$  depende de  $\tau$  y de las propiedades de la función objetivo.

De esta forma, la ecuación (6.17) nos proporciona una cota superior para el error de aproximación de una función  $f$  utilizando una aproximación de  $n$  términos, a partir de un diccionario  $\mathbb{D}$  generado por una base ortonormal  $B$ . En consecuencia, el error de aproximación  $\sigma_n(f, \mathbb{D})_{\mathcal{H}}$  depende tanto de la suavidad de la función  $f$  (medida por  $|f|_{\mathcal{K}_\tau(\mathbb{D})}$ ) como del número de términos empleados en la aproximación.

Finalmente, estamos interesados en verificar si la ecuación (6.17) es válida para diccionarios más generales. El siguiente resultado nos proporciona una cota superior para el error de aproximación en estos casos, extendiendo los resultados obtenidos en el caso de diccionarios generados por bases ortonormales.

**Teorema 6.23.** *Sean  $\mathbb{D}$  un diccionario de funciones,  $\alpha \geq 1/2$  y  $1/\tau = \alpha + 1/2$ . Si  $f \in K_\tau(\mathbb{D})$ , entonces*

$$\sigma_m(f, \mathbb{D})_{\mathcal{H}} \leq C|f|_{K_\tau(\mathbb{D})}m^{-\alpha} \quad \forall m \in \mathbb{N}, \quad (6.18)$$

donde  $C$  depende de  $\tau$ .

*Demostración.* Es suficiente probar la desigualdad (6.18) para funciones  $f$  que puedan expresarse como una suma finita  $f = \sum_j c_j g_j$ , con  $g_j \in \mathbb{D}$  y  $\sum_j |c_j|^\tau \leq M^\tau$ . Sin pérdida de generalidad, podemos suponer que los coeficientes  $c_j$  son positivos y están ordenados de manera decreciente.

Definimos la suma parcial de los primeros  $n$  términos como  $s_1 = \sum_{j=1}^n c_j g_j$  y su residuo asociado como  $R_1 = f - s_1 = \sum_{j>n} c_j g_j$ . Ahora,

## 6. Análisis Teórico del Deep Double Descent

$$c_n^\tau \leq \frac{1}{n} \sum_{j=1}^n |c_j|^\tau \leq \frac{M^\tau}{n}, \quad n \in \mathbb{N}.$$

Por tanto, tomando la raíz  $\tau$ -ésima de ambos lados, y dado que los coeficientes están ordenados de manera decreciente, tenemos que  $c_j \leq M \cdot n^{-1/\tau}$  para  $j > n$ , lo que nos lleva a

$$\sum_{j>n} c_j = \sum_{j>n} c_j^{1-\tau} c_j^\tau \leq M^{1-\tau} n^{1-1/\tau} \sum_{j>n} c_j^\tau \leq Mn^{1-1/\tau}.$$

Esto implica que  $R_1 \in \mathcal{K}_1(\mathbb{D}, Mn^{1-1/\tau})$ . A continuación, se enuncia el Teorema 3.2 de [DT96], el cual utilizaremos para finalizar la demostración.

**Teorema 6.24.** *Sea  $f \in \mathcal{K}_1(\mathbb{D}, 1)$ . Entonces, para el algoritmo Greedy Relajado, se cumple la siguiente estimación:*

$$\|f - G_m^r(f)\| \leq \frac{2}{\sqrt{m}}, \quad \forall m \geq 1,$$

donde  $G_m^r(f)$  denota la aproximación de  $f$  obtenida tras  $m$  iteraciones del algoritmo Greedy Relajado.

Si bien la demostración de este resultado se encuentra fuera de los objetivos de este proyecto, su validez nos permite afirmar que

$$\left\| f - G_m^0(f) \right\|_{\mathcal{H}} \leq \|f\|_{\mathcal{K}_1} \cdot m^{-1/2}.$$

A partir de la desigualdad anterior, existe una función  $s_2$ , que es una combinación lineal de a lo sumo  $n$  elementos  $g \in \mathbb{D}$ , tal que

$$\|f - (s_1 + s_2)\| = \|R_1 - s_2\| \leq 2Mn^{1-1/\tau} \cdot n^{-1/2} = 2Mn^{-\alpha},$$

de donde se obtiene la desigualdad (6.18). □

### 6.4.3. Analogía con el Deep Double Descent

A modo de conclusión de este inciso sobre la teoría de la aproximación no lineal, presentaremos a lo largo de esta sección las similitudes que existen entre dicha teoría y el doble descenso, con el objetivo de establecer ciertas analogías que ofrezcan ideas matemáticas subyacentes asociadas a su ocurrencia.

En primer lugar, el objetivo tanto de la aproximación no lineal como del aprendizaje es esencialmente el mismo: **aproximar una función objetivo mediante funciones más simples**. En el contexto de la teoría de la aproximación, se suele asumir que se conocen los valores de ciertos funcionales lineales simples aplicados a la función objetivo. En cambio, en el aprendizaje, dicha función es completamente desconocida y solo se tiene acceso a un conjunto finito de observaciones, es decir, se puede interpretar como que se conoce el valor de la aplicación de un número finito de funcionales de evaluación. Asimismo, en ambos contextos se recurre al uso de “**procesos no lineales**” para aproximar la función objetivo, lo que permite capturar comportamientos complejos que no podrían representarse adecuadamente mediante aproximaciones provenientes de espacios lineales.

La base a partir de la cual se construyen los aproximantes también desempeña un papel fundamental. Aunque en el caso de utilizar redes neuronales lo que se usa propiamente es

un diccionario, este diccionario será dependiente de la base elegida. De esta manera, cuando llevemos a cabo los experimentos relacionados con la aproximación polinómica, veremos cómo la elección de la base influye de manera significativa en los resultados obtenidos, tal como se discutió en la Subsección 6.4.2.1. En consecuencia, corroboraremos que ciertas bases resultan más adecuadas para aproximar determinadas funciones.

De este modo, se puede intuir que los modelos obtienen sus aproximantes a partir de la base elegida hasta alcanzar el umbral de interpolación. Precisamente en ese umbral, el aproximante queda determinado de forma única por el uso completo de la base, lo cual explica que su error de generalización no esté controlado y pueda alcanzar valores elevados.

Por otra parte, al superar el umbral de interpolación, la base pasa a funcionar como un diccionario, es decir, un conjunto redundante de funciones en el que, a medida que aumenta el número de parámetros del modelo, se incorporan nuevas posibilidades de aproximación. Aunque estas funciones adicionales son redundantes, contribuyen a mejorar la capacidad de generalización del modelo, ya que al contar con un mayor número de posibles soluciones, el nuevo conjunto de hipótesis resulta más adecuado para resolver el problema. De hecho, como se observó en la Subsección 6.4.2.2, la función objetivo tiende a verse más “suave” a medida que aumenta el diccionario. Finalmente, la elección de una “buena” hipótesis de todas las disponibles en el diccionario vendría dada por algún sesgo inductivo del algoritmo de optimización utilizado, tal y como se comentó en la Subsección 6.3.1.

## 6.5. Discusión final

En esta sección hemos expuesto de forma teórica y detallada el *Deep Double Descent*, unificando el enfoque clásico con la práctica moderna a partir de la noción de complejidad efectiva como medida de complejidad del modelo. A partir de esta definición, se identifican las distintas regiones de comportamiento de un modelo durante el entrenamiento, siendo de especial interés la región sobreparametrizada, en la cual hemos centrado nuestro análisis.

Al mismo tiempo, a partir de la descomposición del error en la suma de sesgo y varianza, se introducen las distintas configuraciones que puede adoptar el error de un modelo. Entre ellas, destaca el fenómeno del doble descenso, el cual surge debido a que el sesgo y la varianza predominan en diferentes regiones del comportamiento del modelo, siendo la varianza la que presenta un comportamiento unimodal. También destacamos que, en la región sobreparametrizada, el conjunto de hipótesis disponibles es mayor. De este modo, el verdadero desafío para comprender el fenómeno reside en entender por qué el algoritmo de aprendizaje selecciona ciertas soluciones dentro de ese conjunto sobre otras, y por qué las hipótesis elegidas tienden a generalizar bien.

A partir de un ejemplo de regresión, se introduce un análisis intuitivo del DDD, en el que se observa que su aparición está estrechamente vinculada al término de la varianza. En particular, se verifica que el máximo del error se produce cerca del umbral de interpolación, debido a la configuración de los valores propios que surgen al resolver el problema de regresión asociado. Ligado a esto, se analiza la convergencia del descenso de gradiente, demostrando que, bajo ciertas hipótesis, dicho algoritmo converge hacia soluciones que son minimizadoras globales en la región sobreparametrizada. Esta propiedad se observa tanto en problemas de regresión como en escenarios con datos separables.

A continuación, se presenta la región sobreparametrizada desde el punto de vista de la optimización. En esta sección, se evidencia la existencia de múltiples soluciones dentro de dicha región, así como la ausencia de convexidad local de la función de pérdida en

## 6. Análisis Teórico del Deep Double Descent

cualquier entorno de dichas soluciones. Igualmente, se destaca que, mientras en la zona infraparametrizada las soluciones aparecen de forma aislada, en la zona sobreparametrizada estas conforman variedades continuas.

Posteriormente, se introduce una propiedad más débil que la convexidad, la cual debe cumplir la función de pérdida para garantizar que el GD converja hacia un mínimo global en la región sobreparametrizada. A pesar de conocer la existencia de múltiples soluciones en la región moderna, es necesario entender por qué el algoritmo de aprendizaje tiende a seleccionar aquellas que generalizan adecuadamente. Para ello, se introduce, de forma informal, un concepto de suavidad funcional que orienta al modelo hacia hipótesis más simples, en línea con la filosofía de Ockham. De este modo, el algoritmo de aprendizaje induce un sesgo inductivo suave que lo lleva a concentrarse en subconjuntos óptimos dentro del espacio de hipótesis, lo que contribuye a reducir la varianza en esta región. En última instancia, se presentan los marcos de generalización aplicables en esta región, los cuales se basan en la preferencia de ciertas hipótesis sobre otras.

Para finalizar esta sección, se introducen conceptos básicos de la teoría de la aproximación no lineal, vinculándolos con el dilema del aprendizaje. A su vez, se presentan ciertas analogías con el doble descenso, mostrando cómo el uso de diccionarios con funciones redundantes permite obtener mejores aproximaciones de la función objetivo. Asimismo, se detallan nociones de suavidad en dichos espacios, reforzando la conexión entre la estructura del modelo y su capacidad de generalización.

Para cerrar este análisis, se presenta la Tabla 6.1 como resumen de lo que ocurre en las distintas zonas de actuación de un modelo.

	Régimen clásico (infraparametrizado)	Régimen moderno (sobreparametrizado)
Conjunto de hipótesis	Conjunto subóptimo debido a un sesgo restrictivo	Conjunto óptimo debido a un sesgo inductivo suave
Curva de generalización	Forma clásica de "U"	Descendiente
Paisaje de optimización	Localmente convexo Mínimos locales aislados	No localmente convexo Múltiples mínimos globales formando variedades continuas Cumple la condición $\mu$ -PL
Modelo óptimo	<i>Sweet spot</i> (mínimo de la curva "U") (difícil de conseguir)	Cualquier minimizador global (fácil de encontrar)
Convergencia del GD	GD converge a un mínimo local	GD converge a un mínimo global
Marcos de generalización	Dimensión VC Complejidad de Rademacher	Límite de hipótesis numerable PAC-Bayes
Aproximación no lineal	Uso de bases (simplicidad/eficiencia) para la aproximación	Uso de diccionarios (redundancia) para la aproximación

Tabla 6.1.: Resumen comparativo de las diferencias entre el paradigma clásico y el paradigma moderno.

## 7. Análisis Empírico del Deep Double Descent

En este capítulo abordaremos, de manera práctica, el *Deep Double Descent*. Comenzaremos con su estudio en ejemplos sencillos de regresión, lo cual nos permitirá obtener una primera visión clara e intuitiva de su comportamiento. A continuación y utilizando redes neuronales, se presentarán los principales tipos identificados por la comunidad científica. Asimismo, se explorarán algunos casos adicionales que, si bien no han sido tan estudiados, poseen una relevancia significativa en su análisis.

### 7.1. Materiales y métodos

#### 7.1.1. Datasets

En la parte práctica de este proyecto se emplearán diversos conjuntos de datos (*datasets*) etiquetados, pues nos movemos bajo el enfoque del aprendizaje supervisado. Dado que la experimentación se limita a este tipo de aprendizaje, se han seleccionado *datasets* ampliamente reconocidos y utilizados en problemas de clasificación de imágenes, lo que también nos ayudará a comparar los resultados obtenidos con los de la comunidad científica. A continuación, se detallarán los principales conjuntos de datos utilizados en este estudio: MNIST, CIFAR10 y CIFAR100.

Cabe destacar que, aunque en este proyecto también se utilizan *datasets* sintéticos, estos no serán mencionados en esta sección, pues se detallarán en el propio experimento, y la descripción se centrará exclusivamente en los conjuntos de datos comentados anteriormente.

##### 7.1.1.1. MNIST

El *dataset* MNIST (Modified National Institute of Standards and Technology) [LBBH98] es un conjunto de datos de imágenes ampliamente utilizado para tareas de clasificación. El conjunto consta de 70000 imágenes en escala de grises de dígitos escritos a mano, las cuales se dividen en 60000 imágenes para entrenamiento y 10000 para test. Además, cada imagen posee un tamaño de  $28 \times 28$  píxeles.

##### 7.1.1.2. CIFAR10 y CIFAR100

CIFAR10 [KNHo9] es un conjunto de datos utilizado para tareas de clasificación. Consta de 60000 imágenes en color, distribuidas en 10 clases diferentes, cada una con 6000 imágenes. A su vez, estas imágenes se dividen en 50000 imágenes para entrenamiento y 10000 para test. Cada imagen tiene un tamaño de  $32 \times 32$  píxeles y está en formato RGB (*Red, Green Blue*), es decir, cuenta con 3 capas, cada una asociada con un color del formato RGB.

Por otro lado, CIFAR100 [KNHo9] es un conjunto de datos similar a CIFAR10, pero con 100 clases. Consta de 60000 imágenes en color, divididas en 100 clases de 600 imágenes cada una, con 500 para entrenamiento y 100 para test. Las imágenes tienen un tamaño de  $32 \times 32$  píxeles en formato RGB.

## 7. Análisis Empírico del Deep Double Descent

Para terminar esta sección, se presenta una tabla a modo de resumen de los *datasets* que se utilizarán en este proyecto. La Tabla 7.1 muestra la información relevante sobre cada uno, incluyendo el número total de imágenes, el tamaño de las imágenes y el número de características (píxeles totales), lo que proporciona una visión general de sus características principales y también facilita su comparación.

Dataset	Nº imágenes	Tamaño imagen	Clases	Entrada
MNIST	70000	28 × 28 píxeles (escala de grises)	10	784
CIFAR10	60000	32 × 32 píxeles (RGB)	10	3072
CIFAR100	60000	32 × 32 píxeles (RGB)	100	3072

Tabla 7.1.: Resumen de los *datasets* utilizados. En la tabla aparecen sus principales características, donde la última columna (Entrada) hace referencia al número total de píxeles de cada imagen.

Aunque los *datasets* elegidos no son excesivamente grandes, con el fin de acelerar el proceso de entrenamiento durante un alto número de épocas, se utilizarán subconjuntos de estos en los experimentos. En particular, cuando se haga uso de un subconjunto, se empleará la siguiente notación:

$$\text{dataset}[\text{train/test}]$$

donde *dataset* se refiere a uno de los conjuntos de datos previamente definidos, *train* indica el número de ejemplos utilizados para entrenamiento, y *test* representa el número de ejemplos en el conjunto de test. Un ejemplo sería MNIST[4000/1000], donde se indica que estamos utilizando un subconjunto de MNIST con 4000 ejemplos de entrenamiento y 1000 para test.

### 7.1.2. Protocolo de validación experimental

Para la validación y estimación del rendimiento de cada modelo se emplea la técnica conocida como *hold-out*. Esta técnica consiste en dividir el conjunto de datos en dos subconjuntos: uno destinado al entrenamiento del modelo y otro para la evaluación de su rendimiento (véase la imagen izquierda de la Figura 7.1). En este enfoque, el conjunto de entrenamiento se utiliza para ajustar el modelo, mientras que el conjunto de test se emplea para evaluar su capacidad de generalización frente a datos no vistos previamente.

Es importante destacar que el método habitualmente preferido para la validación de modelos suele ser la validación cruzada o *K-fold cross validation*. Esta técnica consiste en dividir el conjunto de entrenamiento en *K* subconjuntos o pliegues (véase la imagen derecha de la Figura 7.1), donde en cada iteración uno de estos subconjuntos se utiliza como conjunto de test, mientras que los demás se emplean como conjunto de entrenamiento. Este proceso se repite hasta que cada subconjunto haya sido utilizado como conjunto de test en al menos una iteración. Así, cuando se utiliza un valor de *K* = 1, esta técnica se reduce a un *hold-out*, lo que permite considerarla como una extensión de la misma. De este modo, se obtiene una estimación más robusta del rendimiento del modelo frente a nuevos datos no vistos. Finalmente, el rendimiento del modelo se evalúa en el conjunto de test final, que no ha sido utilizado durante ninguna de las particiones de entrenamiento.

Por tanto, la elección del método *hold-out* se debe a su menor requerimiento de potencia computacional y tiempo, lo cual resulta fundamental, dado que nuestros experimentos son especialmente costosos en ambos aspectos (como se puede constatar en las tablas resumen

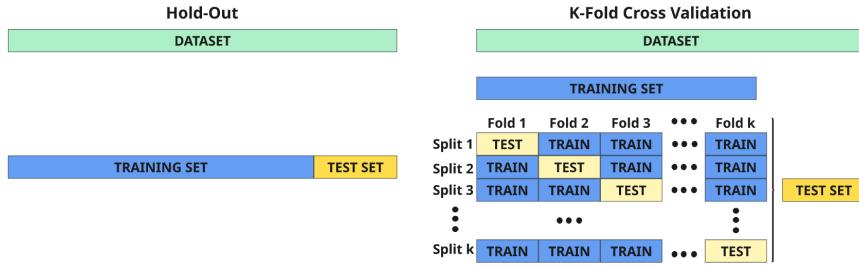


Figura 7.1.: Protocolos de validación experimental. A la izquierda, se visualiza la técnica *hold-out*, en la que el conjunto de datos se divide en dos subconjuntos: uno para el entrenamiento y otro para test. A la derecha, se presenta *K-fold cross validation*, donde el conjunto de entrenamiento se divide en  $K$  pliegues, siendo uno de ellos utilizado como conjunto de test.

incluidas en la parte experimental). Las particiones utilizadas para generar los conjuntos de entrenamiento y test de los distintos conjuntos de datos son las predeterminadas que vienen con estos y que se han descrito en la sección anterior.

### 7.1.3. Arquitecturas utilizadas

En esta sección se presentan las tres arquitecturas principales que han sido diseñadas y utilizadas para abordar la tarea de clasificación propuesta por los *datasets* comentados en la sección anterior. Las arquitecturas seleccionadas varían en términos de su complejidad (número de parámetros) y del tipo de capas que incorporan, desde modelos simples hasta configuraciones más profundas. Este enfoque permitirá analizar el impacto del *Deep Double Descent* en los distintos casos que puede presentarse y sobre las distintas arquitecturas. A continuación, se describen en detalle las arquitecturas implementadas, justificando su elección y su relevancia en el contexto de este estudio.

#### 7.1.3.1. 2NN

Como primer modelo se propone una arquitectura muy simple, formada únicamente por dos capas densas (un perceptrón multicapa (MLP) formado por dos capas). Para ello, primeiramente se añade una capa de aplanamiento (*flattening*) con el objetivo de convertir las imágenes proporcionadas en la entrada en un vector de píxeles unidimensional. A continuación, se añade la primera capa densa, cuya entrada está compuesta por el vector unidimensional anterior y su salida será un número variable de neuronas, todas ellas conectadas con cada entrada del vector. Finalmente, la última capa densa conectará las neuronas de salida de la primera capa densa con un número de neuronas que será igual al número de clases del conjunto de datos con el que estemos trabajando (véase Figura 7.2).

El diseño y elección de este modelo se basa en su relativa sencillez, lo que lo convierte en una opción ideal para llevar a cabo una amplia variedad de experimentos, ya que sus tiempos de entrenamiento serán significativamente más bajos en comparación con otras arquitecturas utilizadas, permitiéndonos estudiar el DDD sin un alto coste computacional. No obstante, dado que los MLPs no cuentan con el *inductive bias* adecuado para trabajar con imágenes, se espera obtener peores resultados en comparación con el uso de CNNs.

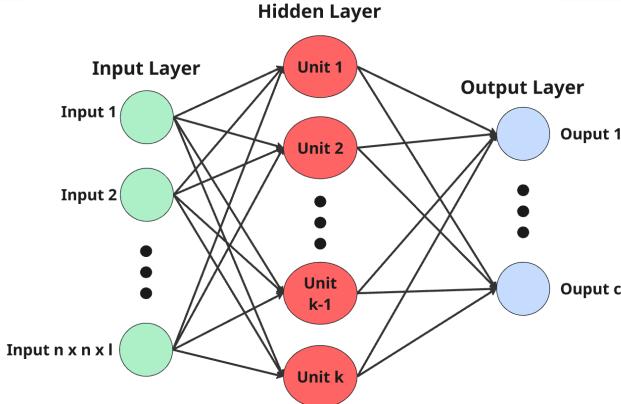


Figura 7.2.: Arquitectura 2NN. La capa de entrada está formada por un número de neuronas igual al número de píxeles de la imagen de entrada. Estas neuronas estarán conectadas a todas las neuronas de la capa oculta, y, a su vez, estas últimas estarán conectadas a todas las neuronas de la capa de salida, la cual corresponde al número de clases del problema.

Finalmente, se expone una tabla con el número de parámetros que conforman esta arquitectura. Este detalle es especialmente relevante para los experimentos, ya que el número de parámetros reflejará el nivel de complejidad de la arquitectura (véase Tabla 7.2).

Entrada	Nº clases	Nº unidades	Nº parámetros
Imagen de tamaño $n \times n \times l$	c	k	$(n^2 \times l) \times k + k + c \times k + c$
28 × 28 × 1	10	50	39760
		100	79550
32 × 32 × 3	10	50	118160
		100	236310
	100	50	122750
		100	245400

Tabla 7.2.: Resumen de las arquitecturas 2NN. Se muestra el número de parámetros dada una determinada imagen de entrada, donde  $k$  hace referencia al número de neuronas de salida de la primera capa densa y  $c$  al número de clases del problema de clasificación. Además, se presentan algunos ejemplos de configuraciones.

### 7.1.3.2. 3CNN

Como segunda arquitectura, se propone una CNN compuesta por 3 capas convolucionales (denominada 3CNN), seguidas de capas de *pooling* para reducir dimensionalidad y, finalmente, una capa densa de salida.

Este modelo se presenta como una opción intermedia en términos de complejidad entre el modelo expuesto previamente y el siguiente modelo que se presentará, donde el número de

filtros en cada capa se ajustará según un parámetro  $k \in \mathbb{N}$ , siguiendo la secuencia  $[k, 2k, 4k]$  (véase Figura 7.3).

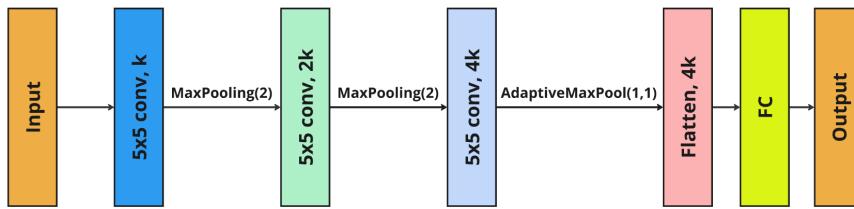


Figura 7.3.: Arquitectura 3CNN. Podemos observar los 3 bloques convolucionales (distinguidos por color), cada uno de ellos con su correspondiente número de filtros ( $[k, 2k, 4k]$  con  $k \in \mathbb{N}$ ) y las correspondientes capas de *pooling*.

Finalmente y dado que el número de parámetros asociados a esta arquitectura es tedioso de indicar (debido a la cantidad de capas que la conforman), se exponen en la Tabla 7.3 los principales valores del parámetro  $k$  (número de filtros) utilizados en el desarrollo del proyecto, lo que permite realizar una comparación con la arquitectura 2NN anterior y con la siguiente. Cabe destacar que la última capa de *pooling* es adaptativa, lo que permite que la salida tenga dimensiones fijas independientemente del tamaño de la imagen de entrada. No obstante, el número de parámetros experimentará una variación mínima al utilizar imágenes en formato escala de grises o en RGB, puesto que el número de capas permanecerá inalterado.

Valor de $k$	Número de parámetros
20	$\approx 103 K$
45	$\approx 512 K$
64	$\approx 1,03 M$

Tabla 7.3.: Número de parámetros de las arquitecturas 3CNN. Se muestra el número total de parámetros para una entrada de tamaño  $32 \times 32 \times 3$  (RGB), considerando distintos valores del parámetro  $k$  y para 10 clases de salida.

### 7.1.3.3. ResNet18 modificada

ResNet18 es una de las variantes de la famosa arquitectura de redes neuronales convolucionales ResNet (*Residual Networks*), propuesta por Kaiming He y su equipo en 2015 [HZRS15]. Esta arquitectura fue diseñada con el objetivo de resolver el problema de la degradación del rendimiento a medida que aumentaba la profundidad de las redes neuronales. ResNet introduce el concepto de conexiones residuales, que permiten que el flujo de información pase a través de capas sin ser afectado por los gradientes de las capas anteriores (véase Apéndice D).

ResNet18, como su nombre indica, tiene 18 capas de profundidad. Su arquitectura presenta varios bloques de convolución seguidos de conexiones residuales, que permiten que la entrada de cada bloque se sume a la salida de dicho bloque. La primera capa convolucional utiliza un filtro de tamaño  $7 \times 7$  y, acto seguido, aparecen los bloques convolucionales formados por 4 capas convolucionales idénticas, donde cada bloque está formado por dos capas convolucionales con conexiones residuales conectadas con la salida de la segunda

## 7. Análisis Empírico del Deep Double Descent

capa convolucional. Finalmente, aparece una capa densa que será la salida de la red (véase Figura 7.4 para más detalle).

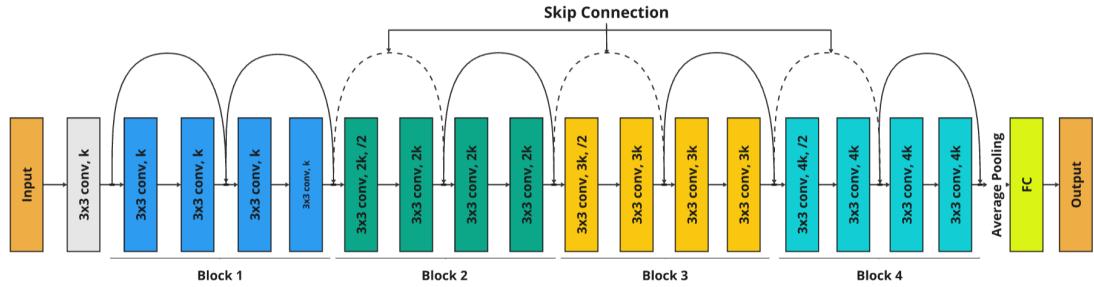


Figura 7.4.: Arquitectura ResNet18 modificada. Podemos observar los 4 bloques convolucionales (distinguidos por color), cada uno de ellos con su correspondiente número de filtros ( $[k, 2k, 3k, 4k]$  con  $k \in \mathbb{N}$ ), junto con las conexiones residuales (flechas continuas) y, finalmente, la capa densa de salida (FC).

La arquitectura que usaremos durante el transcurso de los experimentos (PreActResNet18) es una modificación de la arquitectura ResNet18 original, dado que en la arquitectura original el número de filtros usados en cada bloque convolucional es  $[64, 128, 256, 512]$ , mientras que en nuestro caso será un número variable  $k \in \mathbb{N}$ , con el propósito de crear “distintas” arquitecturas. Para ello, usaremos como número de filtros para cada bloque convolucional la secuencia:  $[k, 2k, 3k, 4k]$  [NKB<sup>+</sup>19], con el objetivo de comparar los resultados obtenidos con los de la literatura existente. Destacamos que, para el valor  $k = 64$ , nuestra arquitectura es la propia arquitectura ResNet18 original. Además, de manera similar a la arquitectura anterior, se utiliza *pooling* adaptativo en la última capa para preservar un tamaño de entrada común a la capa densa, independientemente del tamaño de entrada a la red.

Finalmente, y al igual que en la arquitectura anterior, se presenta una tabla con los principales valores del parámetro  $k$  utilizados en el desarrollo del proyecto (véase Tabla 7.4), con el propósito de permitir una comparación directa de ambas arquitecturas.

Valor de $k$	Número de parámetros
20	$\approx 1.1 \text{ M}$
45	$\approx 5.5 \text{ M}$
64	$\approx 11.1 \text{ M}$

Tabla 7.4.: Número de parámetros de las arquitecturas ResNet18 modificadas para distintos valores de  $k$ , con imágenes de tamaño  $32 \times 32 \times 3$  y 10 clases de salida.

### 7.1.4. Hiperparámetros

Los hiperparámetros que serán utilizados a lo largo de la experimentación, por lo general, son los valores por defecto que vienen con las implementaciones estándar de Pytorch [PGMa19] de los distintos métodos utilizados. Las únicas modificaciones que se han realizado son aquellas necesarias para replicar experimentos descritos en la literatura existente y, en dichos casos, se indicarán específicamente en el propio experimento.

En cuanto al tamaño de lote (*batch size*), variará entre 128 y 256, siendo este último el valor por defecto, de manera similar a la literatura existente, lo que representa un valor relativamente alto, con el objetivo de acelerar el entrenamiento. Respecto a la tasa de aprendizaje del optimizador utilizado (Adam en nuestro caso<sup>1</sup>), se empleará su tasa por defecto. El uso de distintos valores para *batch size* y *learning rate* en un problema donde se manifiesta el doble descenso se pueden observar en el Apéndice B, donde se muestra que, si el fenómeno ocurre, el tamaño del lote utilizado y la tasa de aprendizaje no afectan significativamente al comportamiento del mismo. Finalmente, el número de épocas por defecto son 1000.

Por otra parte, cabe destacar que, salvo que se indique lo contrario, no se introduce ninguna técnica de regularización en los experimentos realizados, debido a que el *Deep Double Descent* se observará sin la influencia de las mismas. De este modo, se pretende estudiar cómo se manifiesta bajo condiciones más puras, sin modificaciones que podrían alterar sus efectos. Cualquier variación en este aspecto será especificada de manera explícita en el propio experimento. Adicionalmente, como función de pérdida a minimizar se utilizará la entropía cruzada<sup>2</sup>, dado que estamos trabajando con problemas de clasificación multiclase.

## 7.2. Experimentos

En esta sección se presentará una batería de resultados obtenidos a partir de los experimentos realizados, los cuales incluyen tanto resultados favorables como desfavorables, con el propósito de proporcionar una visión lo más completa posible de los mismos. La Tabla 7.5 sirve como introducción a los distintos experimentos que se desarrollarán en esta sección.

Todo el código desarrollado, así como los distintos resultados obtenidos, pueden encontrarse en el GitHub <https://github.com/jantonioruiz/Deep-Double-Descent>. De esta manera, se combinan Git y GitHub con el propósito de llevar un control de versiones.

### 7.2.1. Entornos de desarrollo y ejecución

Como preámbulo a los distintos experimentos que se detallan a continuación, se presentan las herramientas utilizadas para su realización. El lenguaje de programación empleado es *Python*, en su versión 3.10, debido a su gran facilidad y versatilidad para trabajar con modelos de aprendizaje profundo. Asimismo, se hace uso de las bibliotecas *PyTorch* junto con las necesarias de *CUDA* para el entrenamiento de las diferentes arquitecturas, así como *NumPy* y *Pandas* para el procesamiento de datos, y *Matplotlib* y *Seaborn* para la generación de gráficos.

Por otro lado, se proporcionan *notebooks* de *Google Colab* con el código asociado a los experimentos que no requieren el uso de redes neuronales y que no demandan grandes recursos. Para los experimentos que requieren el uso de redes neuronales (y que son tan costosos que no se pueden realizar en *notebooks*), se proporciona el código en formato *Python* junto con un *script* de ayuda para lanzar cualquier experimento realizado. Además, para este caso, se incluye en otro *notebook* el código necesario para visualizar las gráficas correspondientes a los resultados obtenidos por los modelos neuronales, con el propósito de evitar tener que ejecutar ningún proceso adicional para su visualización.

---

<sup>1</sup>No obstante, aunque se utilice Adam como optimizador debido a su versatilidad, resultados similares se obtienen para SGD, como se puede observar en [NKB<sup>+</sup>19].

<sup>2</sup> $\mathcal{L}(f(x), g(x)) = -\sum_{i=1}^C f_i(x) \log(g_i(x))$ , donde C es el número de clases,  $f(x)$  la distribución verdadera y  $g(x)$  la predicción del modelo para la entrada  $x$ .

## 7. Análisis Empírico del Deep Double Descent

Experimento	Descripción	Subsección
Aproximación polinómica	Comenzamos empleando aproximación polinómica, con el objetivo de disponer de resultados preliminares que nos permitan concluir si somos capaces de reproducir la aparición del doble descenso.	7.2.2
Noise-wise double-descent	Analizaremos el impacto del ruido tanto en su aparición como en la localización del umbral de interpolación.	7.2.3
Sample-wise double-descent	Estudiaremos el impacto del número de ejemplos en la localización del umbral de interpolación, tratando de mostrar cómo existe una región en la que entrenar con más ejemplos empeora el rendimiento.	7.2.4
Model & Epoch-wise double-descent	Exploraremos en profundidad el <i>Deep Double Descent</i> asociado a la complejidad del modelo y al número de épocas, utilizando los diversos conjuntos de datos y arquitecturas.	7.2.5
Width vs Depth double-descent	Investigaremos el fenómeno al usar redes de mayor profundidad en lugar de mayor anchura, con el propósito de evaluar su robustez frente a la arquitectura empleada.	7.2.6

Tabla 7.5.: Resumen de las ideas principales de los experimentos realizados.

El proceso de ejecución de los experimentos que utilizan redes neuronales se lleva a cabo en los nodos de cómputo (GPUs) de los servidores de la Universidad de Granada (UGR), a los cuales se accede de forma remota mediante SSH. Para el envío de los experimentos, se emplea un *script de Shell* en el que se especifican tanto los parámetros del experimento como el nodo a utilizar. El servidor hace uso de SLURM como gestor de colas, siendo este el encargado de reservar los recursos necesarios en la partición “dios” del servidor. Finalmente, se utiliza la herramienta *Conda* para gestionar tanto los entornos como los paquetes, asegurando que se disponga de todas las bibliotecas necesarias.

Dado que nuestra experimentación es excesivamente costosa, hacemos uso de tres nodos de cómputo, de manera indistinta, dentro de la partición “dios” de los servidores. Las especificaciones de estos nodos se detallan a continuación:

1. Nodo **atenea**: Cuenta con dos procesadores Intel Xeon E5-2630, 128GB de memoria RAM DDR4 y cuatro tarjetas gráficas GTX Titan Xp.
2. Nodo **hera**: Cuenta con dos procesadores Intel Xeon 4114, 128GB de memoria RAM DDR4 y cuatro tarjetas gráficas Titan RTX.
3. Nodo **dionisio**: Cuenta con dos procesadores Intel Xeon Silver 4216, 512GB de memoria RAM DDR4 y dos tarjetas gráficas Quadro RTX 8000.

Aunque estos nodos cuentan con especificaciones diferentes, ofrecen resultados similares en cuanto a tiempo de ejecución. Este hecho permite que puedan ser utilizados de manera

paralela para acelerar los cálculos, ofreciendo tiempos de ejecución significativos para la comparación de resultados.

Finalmente, cabe destacar que, a mediados del mes de abril, se comunicó que el uso de las GPUs de la Universidad de Granada quedaba limitado a un máximo de 12 horas de ejecución continua. Esta limitación a nivel de hardware complica la realización de algunos experimentos, dado que, en este proyecto, es necesario llevar el entrenamiento de los modelos mucho más allá de este tiempo. De este modo, los experimentos realizados tras esta comunicación presentan configuraciones de parámetros distintas en comparación con el resto. Es importante resaltar que la batería experimental llevada a cabo en este proyecto es de gran calibre, lo que conlleva una mayor demanda de recursos y tiempo de ejecución para obtener resultados más precisos y concluyentes. A modo ilustrativo, en la Tabla 7.6 se presenta un resumen del número total de experimentos ejecutados, así como del total de horas de GPU acumuladas para aquellos experimentos que hacen uso de modelos neuronales<sup>3</sup>.

Nº total experimentos	Nº horas GPU (servidor UGR)	Equivalencia días
40	≈ 1795h	≈ 75d <sup>4</sup>

Tabla 7.6.: Resumen del número total de experimentos, junto con el total de horas de cómputo empleadas en el servidor de la UGR y su equivalencia en días.

### 7.2.2. Aproximación polinómica

Este experimento sirve como preámbulo al resto de experimentos y tiene como objetivo proporcionar una comprensión clara y sencilla de por qué puede manifestarse el *Deep Double Descent* al aumentar la complejidad de un modelo. A través de este análisis preliminar, se busca sentar las bases para explorar más a fondo cómo la complejidad afecta al comportamiento y rendimiento de los modelos.

Para ello, intentaremos aproximar una función objetivo mediante tres enfoques distintos: el primero basado en la base de Legendre, el segundo empleando la base polinómica clásica, y el tercero utilizando una base polinómica redundante, donde en todos los casos usaremos un número finito de puntos muestrados de dicha función objetivo para realizar esa aproximación. Este experimento no empleará redes neuronales, sino que se basará en una simple regresión polinómica, cuya solución obtendremos mediante la pseudoinversa.

Por tanto, el objetivo de este experimento es encontrar la función  $f$  que modele el valor esperado de una variable aleatoria dependiente  $y$  (salida) en términos del valor esperado de una variable aleatoria independiente  $x$  (entrada), es decir,  $f(x) = y$ . Dado que la función objetivo es desconocida, buscamos aproximarla, en este caso, mediante el uso de funciones polinómicas. De esta manera, buscamos aproximar la variable  $y$  utilizando las tres aproximaciones descritas en la Tabla 7.7, cada una construida sobre una base polinómica distinta.

Sabemos que la solución de norma mínima para el problema de mínimos cuadrados asociado  $Xw = y$  viene dada por la pseudoinversa (véase Sección 2.2), donde, dependiendo de la base elegida, la matriz  $X$  viene dada por:

<sup>3</sup>El resto de experimentos, que requieren menor potencia computacional y se ejecutan en notebooks de Google Colab, no se tienen en cuenta en este resumen, ya que el tiempo requerido es despreciable en comparación.

<sup>4</sup>Este tiempo total debe ser dividido entre 3, pues se utilizaron tres nodos de cómputo en paralelo. Por tanto, el tiempo real de ejecución sobre las GPUs de la UGR fue de aproximadamente 25 días.

$$X_{Legendre} = \begin{bmatrix} P_0(x_0) & P_1(x_0) & \cdots & P_n(x_0) \\ P_0(x_1) & P_1(x_1) & \cdots & P_n(x_1) \\ P_0(x_2) & P_1(x_2) & \cdots & P_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ P_0(x_m) & P_1(x_m) & \cdots & P_n(x_m) \end{bmatrix}, \quad X_{Vandermonde} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^n \end{bmatrix},$$

$$X_{Redundant} = \begin{bmatrix} 1 & x_0 & 1+x_0^2 & \cdots & 1+x_0+\cdots+x_0^n \\ 1 & x_1 & 1+x_1^2 & \cdots & 1+x_1+\cdots+x_1^n \\ 1 & x_2 & 1+x_2^2 & \cdots & 1+x_2+\cdots+x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & 1+x_m^2 & \cdots & 1+x_m+\cdots+x_m^n \end{bmatrix},$$

donde  $X_{Legendre}$ ,  $X_{Vandermonde}$  y  $X_{Redundant}$  son las matrices  $X$  relativas, respectivamente, a la base de polinomios de Legendre, a la base polinómica clásica y a la base polinómica redundante. Así, el vector de parámetros viene dado por  $w = X^\dagger y$ , donde  $X^\dagger$  es la pseudoinversa de la matriz  $X$  asociada a cada aproximación.

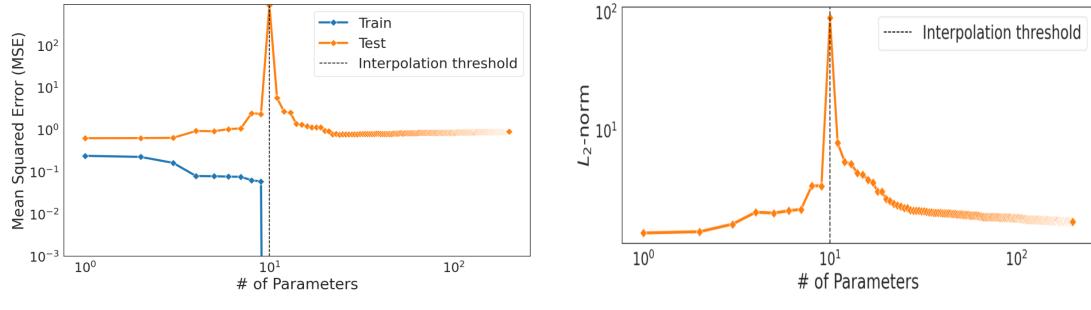
Base	Aproximación
Legendre	$w_0 \cdot P_0(x) + w_1 \cdot P_1(x) + w_2 \cdot P_2(x) + \cdots + w_n \cdot P_n(x) = y$
Clásica	$w_0 \cdot 1 + w_1 \cdot x + w_2 \cdot x^2 + \cdots + w_n \cdot x^n = y$
Redundante	$w_0 \cdot 1 + w_1 \cdot (1+x) + w_2 \cdot (1+x+x^2) + \cdots + w_n \cdot (1+x+\cdots+x^n) = y$

Tabla 7.7.: Aproximaciones polinómicas de grado  $n$  utilizadas para regresión polinomial para modelar la relación entre la variable aleatoria independiente  $x$  y la variable aleatoria dependiente  $y$ . Se presentan tres bases diferentes: la base de Legendre, donde  $P_i(x)$  hace referencia al  $i$ -ésimo polinomio de Legendre<sup>5</sup>; la base polinomial clásica, que utiliza una expansión estándar de potencias de  $x$ , y la base redundante, que incrementa la complejidad del modelo mediante combinaciones de términos de grados inferiores. En todas ellas, el vector de parámetros  $w = [w_0, w_1, \dots, w_n]$  se aprende durante la fase de entrenamiento, utilizando la pseudoinversa.

### 7.2.2.1. Función lineal

En primera instancia, buscamos aproximar la función objetivo dada por  $f(x) = 2x + \cos(25x)$  utilizando 10 puntos muestrados de esa función y usando la pseudoinversa, con el objetivo de replicar y dar más contexto al experimento presentado en [SKR<sup>+</sup>23]. La tendencia de la función objetivo la marca la función lineal  $2x$  y la función trigonométrica  $\cos(25x)$  nos ayudará a introducir cierto ruido a esa función de tendencia, asimilando el posible ruido que encontramos en el mundo real.

<sup>5</sup>Los polinomios de Legendre forman un sistema de polinomios completos y ortogonales en el intervalo  $[-1, 1]$ .



(a) Doble descenso para la aproximación polinómica de Legendre.

(b) Norma euclídea del vector de parámetros asociado a cada aproximación.

Figura 7.5.: Doble descenso al utilizar aproximación polinómica de Legendre (imagen izquierda), donde el umbral de interpolación corresponde al número de datos de entrenamiento (línea negra punteada), junto con la norma del vector de parámetros asociado (imagen derecha), la cual decrece tras dicho umbral.

Para el caso de la aproximación de Legendre, podemos observar en la Figura 7.5a cómo se manifiesta el doble descenso. De esta manera, al superar el umbral de interpolación, encontrado justo al usar el mismo número de parámetros que datos de entrenamiento, el error en el conjunto de test vuelve a disminuir, produciéndose el segundo descenso.

Por otro lado, podemos observar en la Figura 7.6 las distintas aproximaciones que obtiene el modelo al utilizar distinto número de parámetros. Antes de llegar al umbral de interpolación, el modelo no tiene la suficiente capacidad como para ajustarse a cada uno de los datos. Al alcanzar el umbral de interpolación, el modelo se ve obligado a ajustarse exactamente a cada uno de los puntos de entrenamiento, lo que lo convierte en una aproximación única. En este caso, esta “*rigidez*” del modelo no asegura una buena generalización debido a presentar numerosas oscilaciones. Al superar dicho umbral, el aumento en el número de parámetros permite la existencia de múltiples modelos de interpolación para cada grado, facilitando la selección de una opción que generalice bien. De este modo, el modelo adquiere mayor flexibilidad para aproximar los datos, lo que da lugar a una función cada vez más “suave”.

En la Figura 7.5b se observa cómo el modelo, a medida que aumenta su capacidad, sigue reduciendo la norma del vector de parámetros aprendido, ligado a que, como se explicó en la Sección 2.2, la pseudoinversa proporciona la solución de norma mínima. No obstante, este comportamiento nos indica que el modelo se va “**autoregularizando**” hacia soluciones cada vez más simples (suaves) y que se asemejan más a la solución inicial (véanse la primera y última imágenes de la Figura 7.6), lo cual es coherente con lo expuesto en la parte teórica.

Por otro lado, en las aproximaciones polinómicas clásicas, es posible percibir en la Figura 7.7 que dicho fenómeno no se presenta, al menos con el mismo número de parámetros, incluso habiendo obtenido un error de entrenamiento nulo. Es por esto que las aproximaciones obtenidas por el modelo a partir del umbral de interpolación son peores (véase Figura 7.9). No obstante, una vez superado el umbral de interpolación, estas aproximaciones clásicas, al igual que la aproximación de Legendre, ofrecen soluciones con menor norma del vector de parámetros, como puede notarse en la Figura 7.8. Sin embargo, estas normas parecen estabilizarse en un valor muy elevado, a diferencia de lo observado con la base de Legendre. Esto sugiere que, con estas bases, la regularización implícita del modelo no resulta tan efectiva, lo que podría explicar la ausencia del DDD.

## 7. Análisis Empírico del Deep Double Descent

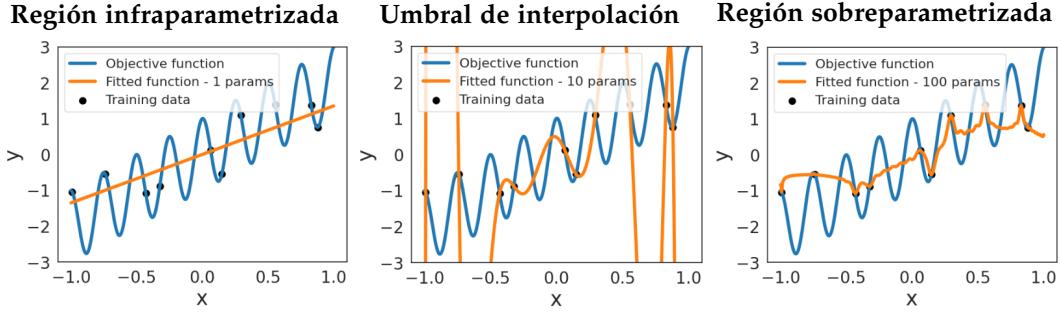
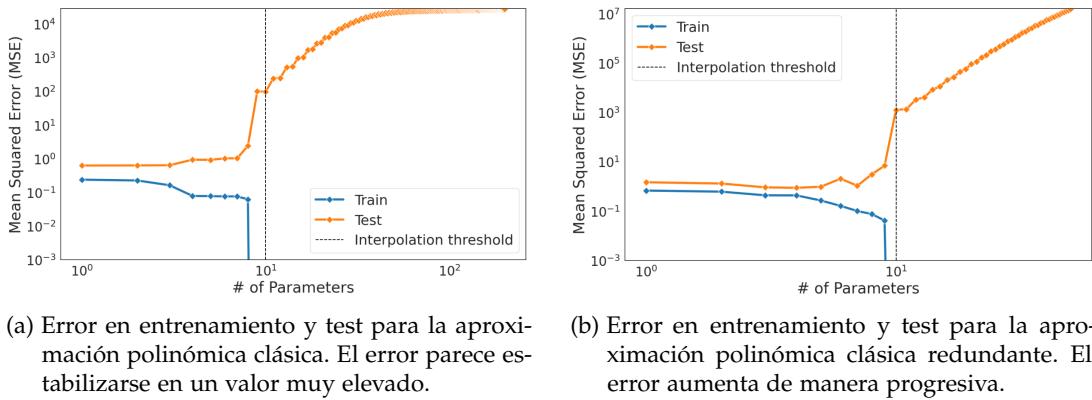


Figura 7.6.: Intuición del *Deep Double Descent* usando regresión polinómica. Cuando nos encontramos en la región infraparametrizada, el modelo no es capaz de capturar todos los datos de entrenamiento, haciendo que el *bias* del modelo sea grande, aunque la varianza sea pequeña. En el umbral de interpolación, el modelo captura perfectamente todos los datos, haciendo que el *bias* sea pequeño pero la varianza sea grande, pues la función aprendida dependerá de la posición de los datos de entrenamiento. Finalmente, en la región sobreparametrizada, el modelo está regularizado hacia una solución de norma pequeña similar a la inicial.

Vale la pena mencionar que la Figura 7.7a difiere notablemente de la Figura 7.7b. En la primera, se observa cómo el error de test tiende a estabilizarse, aunque en un valor considerablemente alto. En la segunda, en cambio, el error de test parece aumentar de forma progresiva (no obstante, si se analiza con detenimiento, este comienza a estabilizarse muy lentamente). De hecho, esta última imagen representa, en gran medida, el claro ejemplo que mantiene la sabiduría convencional sobre el error de generalización, siguiendo la clásica descomposición en sesgo y varianza, lo que constituye un caso desfavorable para el DDD. Es posible que, con un mayor número de parámetros, el error acabe estabilizándose o incluso disminuyendo. Sin embargo, esto no ha podido comprobarse debido a las limitaciones computacionales, dado que el cálculo de la pseudoinversa para esta base resulta demasiado costoso.



(a) Error en entrenamiento y test para la aproximación polinómica clásica. El error parece estabilizarse en un valor muy elevado.

(b) Error en entrenamiento y test para la aproximación polinómica clásica redundante. El error aumenta de manera progresiva.

Figura 7.7.: Error en entrenamiento y test para la aproximación polinómica clásica y la redundante. El error en el conjunto de test se incrementa progresivamente. En ambos casos no se aprecia el DDD, lo que sugiere que la base utilizada es un elemento clave para su aparición.

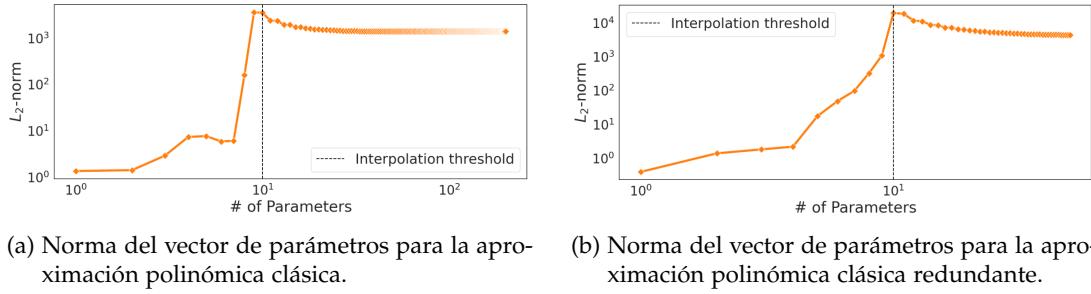


Figura 7.8.: Normas del vector de parámetros para las aproximaciones polinómicas clásicas. A medida que aumenta el número de parámetros, la norma del vector seleccionado tiende a estabilizarse en un valor elevado, lo que indica que la regularización implícita resulta menos efectiva con estas aproximaciones.

Finalmente, y de manera complementaria, se exponen en el Apéndice C los resultados obtenidos al resolver este problema utilizando el descenso de gradiente como método de optimización, donde se observa la misma tendencia previamente descrita.

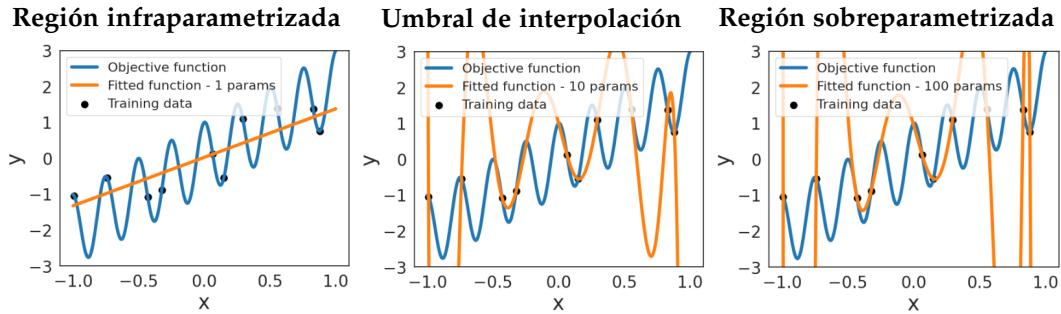


Figura 7.9.: Aproximaciones polinómicas clásicas de la función objetivo. Se observa que, una vez superado el umbral de interpolación, añadir un mayor número de parámetros no contribuye a obtener una mejor aproximación.

La conclusión que podemos obtener de los resultados anteriores es que **la base sobre la que estamos trabajando juega un papel fundamental en las aproximaciones obtenidas**. De hecho, podemos considerar la base de Legendre, en cierta medida, como “redundante”, dado que los polinomios que la conforman incluyen términos de polinomios anteriores. Por otro lado, la base polinomial clásica redundante se incorporó con la expectativa de que se reprodujera el fenómeno, debido también a su redundancia, sin embargo, no se reproduce. Esto sugiere que la base de Legendre, además de ser redundante, es más potente en características que no poseen las otras dos aproximaciones, lo que permite realizar aproximaciones progresivamente más precisas.

### 7.2.2.2. Función hiperbólica

Nos centramos ahora en trabajar con la aproximación de Legendre para una función objetivo cuya tendencia está dada por una función de tipo hiperbólica ( $\tanh(x)$ ), y el ruido será determinado por la misma función que en la subsección anterior ( $\cos(25x)$ ). Este experimento

## 7. Análisis Empírico del Deep Double Descent

tiene como propósito verificar cómo el ruido contribuye a visualizar de manera más precisa el *Deep Double Descent*.

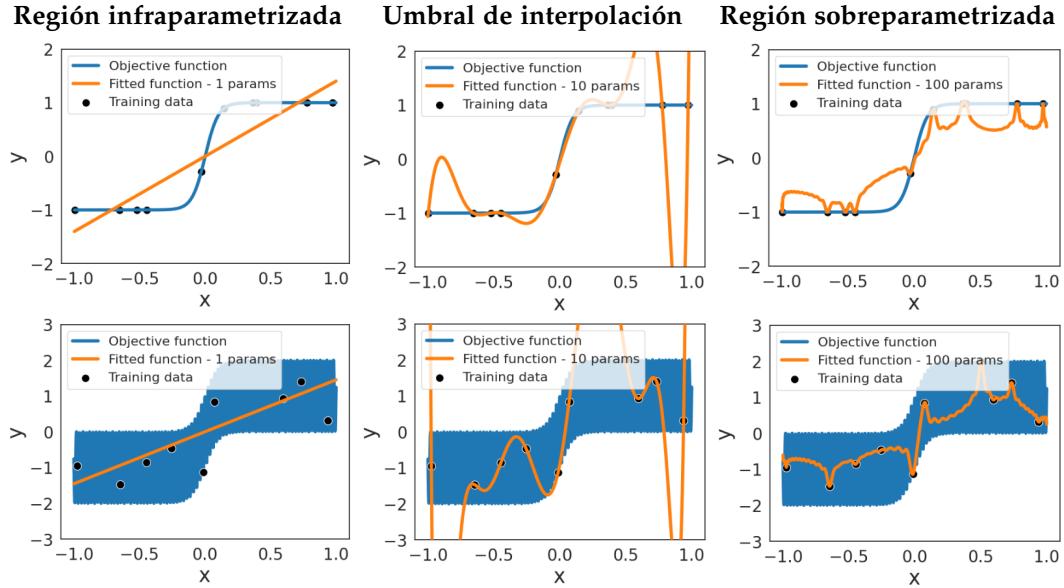


Figura 7.10.: Aproximaciones polinómicas de Legendre de la función objetivo, tanto sin ruido (fila superior) como con ruido (fila inferior). A medida que aumentan los parámetros, la predicción se ajusta a la función objetivo, incorporando también un cierto componente de suavizado o regularización implícita.

Para ello, usamos 10 puntos muestrados de la función objetivo para el entrenamiento. En la Figura 7.10 podemos apreciar las distintas aproximaciones obtenidas por el modelo para el caso sin ruido y el caso con ruido, donde se puede observar cómo el aumento de parámetros ayuda a obtener mejores predicciones.

En la Figura 7.11 podemos observar las curvas del error de test y entrenamiento para las distintas aproximaciones obtenidas en ambos casos. Destacamos el hecho de que, para el caso ruidoso, el doble descenso se presenta de forma más clara, donde al aumentar el número de parámetros se obtienen menores errores en el conjunto de test que los obtenidos durante el primer descenso. Este comportamiento va en consonancia con lo descrito en la literatura científica, donde al introducir ruido se observa el DDD con mayor claridad. Sumado a esto, las gráficas de la función objetivo ruidosa de la Figura 7.10 muestran cómo, al autoregularizarse el modelo y obtener una solución más simple, esta intenta suavizarse en torno al centro del conjunto de datos de entrenamiento, de modo que cualquier dato ruidoso no afecte excesivamente a la aproximación.

### 7.2.2.3. Discusión

Finalmente, para concluir este apartado, se exponen las principales conclusiones derivadas de los experimentos realizados:

- La base utilizada para la aproximación influye en la aparición del doble descenso, manifestándose al emplear la base de Legendre, pero no al utilizar las otras bases.

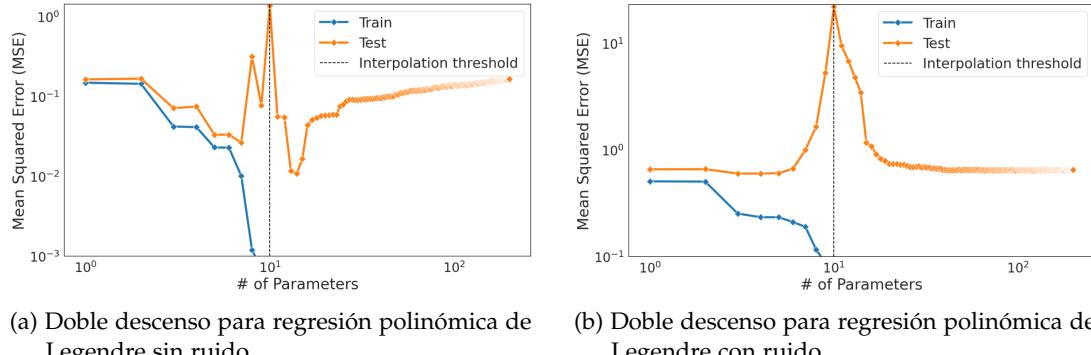


Figura 7.11.: Ejemplos de doble descenso en regresión polinómica de Legendre para una función objetivo, tanto sin ruido como con ruido. En el escenario con ruido, se observa el DDD de forma más clara.

- Al aumentar el número de parámetros utilizados, la solución se “autoregula” hacia la solución de norma mínima del vector de parámetros, haciendo que la aproximación encontrada sea cada vez más suave, siguiendo la filosofía de Ockham.
- El ruido contribuye a obtener representaciones más claras del fenómeno, lo que resulta en errores de generalización menores que los obtenidos en el primer descenso. Esto sugiere que el modelo estaría identificando una forma de aislar ese ruido y centrarse en los datos realmente relevantes.

### 7.2.3. Noise-wise double descent

En esta subsección, y como preámbulo al resto de experimentos, se estudia cómo varía el doble descenso al aumentar el porcentaje de ruido uniforme aleatorio en las etiquetas. De este modo, añadimos este tipo de doble descenso a las variantes clásicas propuestas por Nakkiran et al. en [NKB<sup>+</sup>19].

Para ello, se llevará a cabo un experimento con la arquitectura 2NN sobre el dataset MNIST[4000/1000], modificando el porcentaje de ruido añadido en las etiquetas.

En la Figura 7.12 se observa como **un modelo que no presenta doble descenso sobre el conjunto de datos original sí lo experimenta al agregarle ruido**. Además, en modelos donde si aparece el DDD, el máximo del error de test aumenta al incrementar el porcentaje de ruido, lo que concuerda con lo conjecturado en la literatura científica, ya que el modelo, eventualmente, memorizará dicho ruido.

También se aprecia que, a medida que aumenta el ruido, se requieren más parámetros para alcanzar errores de test más bajos que el mínimo obtenido durante el primer descenso. Este fenómeno, junto con el aumento del tiempo de entrenamiento al incrementar el ruido (véase Tabla 7.8) y, considerando que, en escenarios reales, el porcentaje de ruido no suele ser excesivamente alto, nos lleva a optar por configuraciones con un nivel de ruido en torno al 10 – 20 % para los experimentos restantes.

Finalmente, en la Figura 7.13 podemos verificar cómo, al aumentar el porcentaje de ruido en las etiquetas, el umbral de interpolación se desplaza hacia la derecha, lo que sugiere que los patrones que necesita aprender el modelo son más complejos a medida que aumentamos el ruido, siguiendo una lógica coherente. Además, este hecho pone de manifiesto que, en

## 7. Análisis Empírico del Deep Double Descent

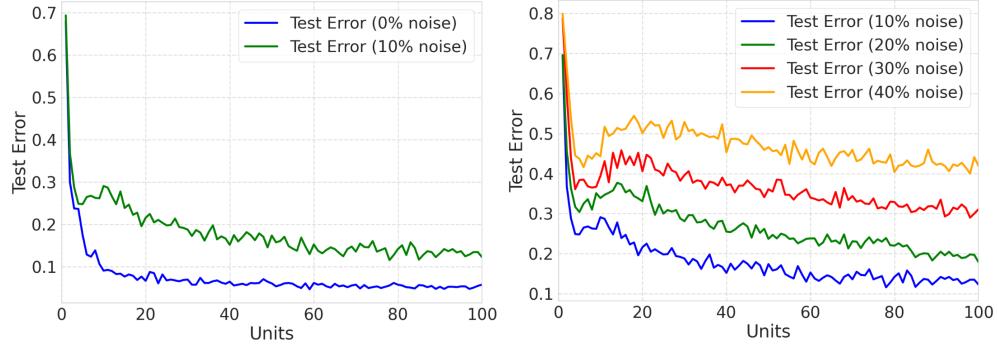


Figura 7.12.: Error en test de la arquitectura 2NN sobre el subconjunto MNIST[4000/1000] para diferente nivel de ruido. A la izquierda, en ausencia de ruido añadido, el *Deep Double Descent* no se manifiesta. En cambio, a la derecha, al aumentar el nivel de ruido, el pico de la curva se vuelve cada vez más pronunciado.

datos multidimensionales como las imágenes, se requieren más parámetros que el número de ejemplos utilizados. Esto es lógico, ya que una imagen no puede ser tratada como un objeto unidimensional, donde el umbral de interpolación permanece inalterado independientemente de la presencia de ruido, debido a que, para casos unidimensionales, siempre es posible memorizar un dato con un único parámetro (véase Figura 7.5).

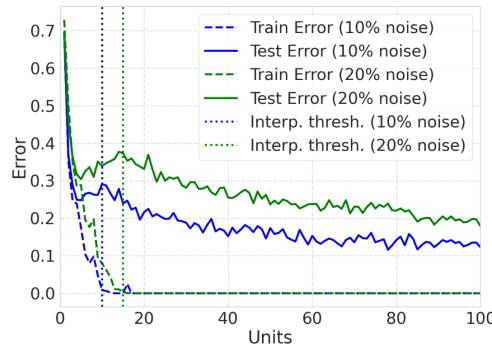


Figura 7.13.: Error en entrenamiento (línea discontinua), test (línea continua) y umbral de interpolación (línea punteada) respecto a la red 2NN sobre el subconjunto MNIST[4000/1000] con distinto nivel de ruido añadido. Al aumentar el nivel de ruido, el umbral de interpolación se desplaza hacia la derecha.

Modelo	Dataset	Ruido en etiquetas	Entrenamiento
2NN (1 – 100)	MNIST[4000/1000]	0 %	26h 10min
2NN (1 – 100)	MNIST[4000/1000]	10 %	26h 50min
2NN (1 – 100)	MNIST[4000/1000]	20 %	27h 23min
2NN (1 – 100)	MNIST[4000/1000]	30 %	28h 8min
2NN (1 – 100)	MNIST[4000/1000]	40 %	30h 14min

Tabla 7.8.: Resumen de los experimentos para el DDD por nivel de ruido.

### 7.2.3.1. Discusión

En conclusión, a partir de los resultados obtenidos en este experimento, se presentan las siguientes conclusiones clave:

- El ruido tiene un impacto significativo en la aparición del *Deep Double Descent*, provocando que se pueda manifestar en conjuntos de datos donde, sin ruido, no ocurre.
- El ruido afecta al máximo error alcanzado, provocando un pico más pronunciado a medida que aumenta el nivel de ruido, relacionado con el hecho de que el modelo memoriza ese ruido.
- El ruido provoca un desplazamiento del umbral de interpolación hacia la derecha, ligado a que los patrones que el modelo debe aprender se vuelven más complejos.

### 7.2.4. Sample-wise double descent

En esta subsección se presentan los experimentos realizados para analizar el impacto del tamaño del conjunto de datos sobre el doble descenso, conocido como *sample-wise double descent*. Este efecto se manifiesta al modificar la cantidad de datos empleados para entrenar un modelo específico. Al comparar las curvas del error de generalización, se observa que **entrenar con un mayor número de ejemplos puede, de manera casi paradójica<sup>6</sup>, empeorar el rendimiento del modelo.**

Dado que la experimentación presentada en la literatura existente resulta excesivamente costosa de replicar<sup>7</sup>, se han diseñado experimentos más accesibles que permiten analizarlo manteniendo un enfoque representativo. La idea para crear estos experimentos sencillos surge del hecho de que si un determinado modelo presenta *model-wise doble descent* para un determinado número de ejemplos de entrenamiento, si aumentamos dicho número de ejemplos, el pico del error de test se desplazará hacia la derecha. Este desplazamiento da lugar a una región específica, a la que denominamos **zona de interés**. Dentro de esta zona se verifica que entrenar con más ejemplos puede empeorar el rendimiento del modelo (véase la zona sombreada en color rojo de la Figura 7.14). Sin embargo, es importante resaltar que, fuera de esta zona, el incremento en la cantidad de ejemplos de entrenamiento suele tener un efecto positivo en el rendimiento del modelo, como nos dice la sabiduría clásica.

Por tanto, para la realización de este experimento se utilizará la red 2NN, donde el número de unidades de salida de la primera capa densa variará desde 1 hasta 200. Además, se utilizará el dataset MNIST, sobre el que se extraerán 4000 y 8000 ejemplos para los distintos experimentos a realizar y a los que se les agregará un ruido del 10 % en sus etiquetas.

Al igual que ocurría al aumentar el nivel de ruido, si aumentamos el número de ejemplos de entrenamiento, el umbral de interpolación también se desplaza hacia la derecha (véase Figura 7.14). Es decir, el modelo requiere de un mayor número de parámetros para memorizar un mayor número de ejemplos de entrenamiento, lo que, en principio, resulta coherente con la complejidad efectiva del modelo y con la propia lógica humana.

Finalmente, se muestra en la Tabla 7.9 el tiempo de ejecución necesario para la realización de este experimento. Cabe destacar que, en el experimento donde se calcula la media de 3

---

<sup>6</sup>En el sentido que, generalmente, en el aprendizaje automático siempre se busca tener la mayor cantidad de datos posible de cara a entrenar.

<sup>7</sup>Nakkiran et al. [NKB<sup>+</sup>19] presentan este tipo de doble descenso utilizando Transformers (Figura 3) como modelo de procesamiento de lenguaje natural, incrementando su capacidad a través de su dimensión de incrustación (*embedding*) y empleando un conjunto de datos de traducción automática.

## 7. Análisis Empírico del Deep Double Descent

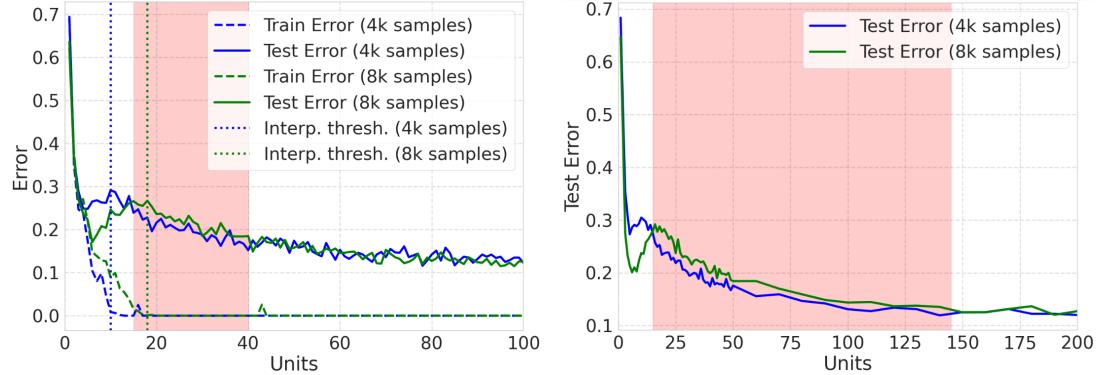


Figura 7.14.: Ejemplo de *sample-wise double descent* para la red 2NN sobre dos subconjuntos de MNIST con 10 % de ruido añadido, utilizando 4000 y 8000 ejemplos de entrenamiento. A la izquierda, aparece el resultado de una única realización del experimento. A la derecha, aparece el resultado obtenido al realizar la media de 3 experimentos. Se puede observar cómo, con el doble de datos (línea verde), no solo el punto de interpolación se desplaza a la derecha (apareciendo más tarde), sino también el error de test es, en la mayor parte de casos, menor cuando se emplean menos ejemplos de entrenamiento.

ejecuciones, se utiliza el modelo 2NN hasta 50 unidades (de 1 en 1) y, a partir de este valor, se suman las unidades de 10 en 10 hasta llegar a 200 unidades, con el propósito de acelerar el entrenamiento fuera de la zona de interés que proporcionaba el primer experimento.

Modelo	Dataset	Entrenamiento
2NN (1 – 100)	MNIST[4000/1000] (10 % label noise)	26h 50min
2NN (1 – 100)	MNIST[8000/1000] (10 % label noise)	59h
2NN (1 – 200) × 3	MNIST[4000/1000] (10 % label noise)	53h 10min
2NN (1 – 200) × 3	MNIST[8000/1000] (10 % label noise)	104h 30min
2NN (1 – 100)	MNIST[12000/1000] (10 % label noise)	65h 50min
2NN (1 – 100)	MNIST[16000/1000] (10 % label noise)	102h 49min

Tabla 7.9.: Resumen de los experimentos para el doble descenso por número de ejemplos de entrenamiento.

### 7.2.4.1. Ratio parámetros/ejemplos

En este apartado, se analiza la relación entre el número de parámetros y el número de ejemplos con el objetivo de estudiar la tendencia del umbral de interpolación en función de este cociente. Para ello, se lleva a cabo un experimento utilizando la arquitectura 2NN sobre el dataset MNIST, variando la cantidad de ejemplos de entrenamiento. Además, se introduce un 10 % de ruido en las etiquetas.

La Figura 7.15 confirma nuevamente que, al aumentar el número de ejemplos de entrenamiento, el umbral de interpolación se desplaza hacia la derecha. Asimismo, este experimento valida que dicho máximo se produce cuando el modelo alcanza un error de entrenamiento cercano a cero, en correspondencia con la noción de complejidad efectiva. Por otra parte, se

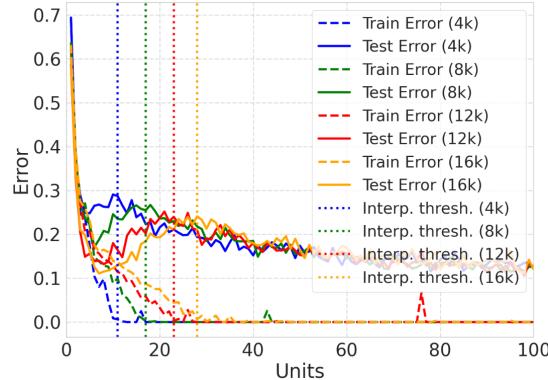


Figura 7.15.: Error en entrenamiento (línea discontinua) y test (línea continua) de la arquitectura 2NN sobre el dataset MNIST[4000 – 16000] y 10 % de ruido añadido a las etiquetas. Además, se muestra el umbral de interpolación (línea punteada) para cada número de ejemplos, observándose cómo dicho umbral se desplaza hacia la derecha y cómo la magnitud del error máximo disminuye a medida que se emplea un mayor número de ejemplos de entrenamiento.

puede observar en dicha gráfica que, al incrementar la cantidad de ejemplos de entrenamiento, el valor máximo del error de test disminuye, posiblemente debido a que el modelo dispone de mayor información, lo que le permite aprender de forma más eficaz y, en consecuencia, cometer menos errores.

No obstante, a partir de este experimento también es posible extraer conclusiones sobre el ratio entre parámetros y ejemplos en el que aparece dicho umbral de interpolación, lo que permite estudiar su comportamiento a medida que ambos aumentan. A partir de los datos presentados en la Tabla 7.10, se observa que el cociente entre el número de parámetros y el número de ejemplos en el que se alcanza el umbral de interpolación tiende a decrecer a medida que aumenta la cantidad de datos de entrenamiento. Este comportamiento sugiere que, en el límite de un número suficientemente grande de ejemplos, dicho ratio se aproxima progresivamente a uno, indicando una relación más equilibrada entre la capacidad del modelo y la cantidad de datos necesarios para alcanzar el régimen de interpolación.

Finalmente, los tiempos de entrenamiento correspondientes a este experimento se muestran en la Tabla 7.9. Se destaca el notable incremento en el tiempo de entrenamiento a medida que aumenta el número de ejemplos, lo que limita la viabilidad de un análisis más exhaustivo sobre la convergencia de este cociente.

Modelo	Número de parámetros	Ejemplos	Ratio
2NN (11)	8755	4000	$\approx 2,19$
2NN (17)	13525	8000	$\approx 1,69$
2NN (23)	18295	12000	$\approx 1,52$
2NN (28)	22270	16000	$\approx 1,39$

Tabla 7.10.: Ratio de parámetros/ejemplos relativo al umbral de interpolación para diferentes cantidades de ejemplos de entrenamiento. A medida que aumentamos el número de ejemplos de entrenamiento, el ratio se approxima a 1.

## 7. Análisis Empírico del Deep Double Descent

### 7.2.4.2. Discusión

Para cerrar esta subsección, se exponen las conclusiones más relevantes alcanzadas:

- Al aumentar el número de ejemplos, el umbral de interpolación se desplaza hacia la derecha, dado que el modelo requiere más capacidad para alcanzar su complejidad efectiva. Esto da lugar a una zona en la que, de manera paradójica, entrenar con un mayor número de ejemplos empeora el rendimiento.
- Se calcula, mediante un ejemplo sencillo, el promedio entre el número de parámetros y los ejemplos en los que aparece el umbral, y se observa que sigue una tendencia decreciente hasta converger, aparentemente, hacia el valor de 1.

### 7.2.5. Model & Epoch-wise double descent

Nos centramos ahora en la verificación del *Deep Double Descent* en sus dos versiones más comunes: el doble descenso en función de la complejidad del modelo (*model-wise*) y en función del número de épocas (*epoch-wise*). A tal efecto, representaremos ambos comportamientos de manera conjunta en una gráfica que muestra el número de parámetros en el eje X y el número de épocas en el eje Y. De este modo, el *model-wise double descent* se puede observar al fijar el eje Y, mientras que el *epoch-wise double descent* se visualiza al fijar el eje X. Además, esta representación permite analizar la tendencia que sigue el umbral de interpolación en función del cociente entre el número de épocas y el número de parámetros.

#### 7.2.5.1. 2NN

En primer lugar, trabajaremos con la arquitectura 2NN debido a su baja capacidad. Para garantizar que esta arquitectura cuente con un número de parámetros superior al de ejemplos y, al mismo tiempo, acelerar el entrenamiento, usaremos subconjuntos de 4000 ejemplos.

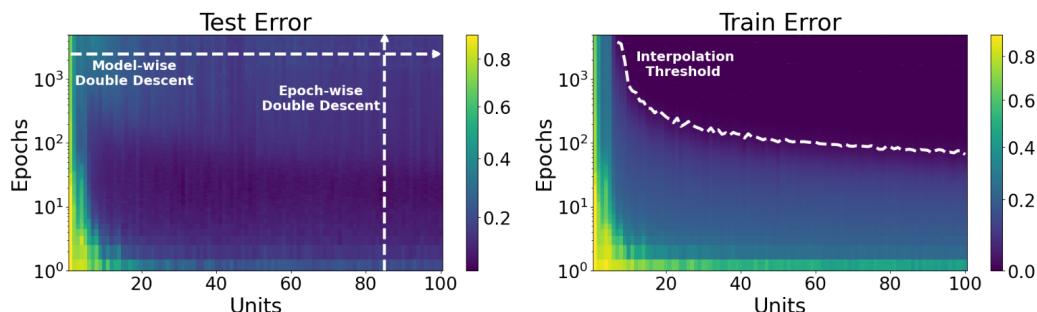


Figura 7.16.: A la izquierda, se muestra el error de test en función del tamaño del modelo y del número de épocas de entrenamiento. A la derecha, se representa el error de entrenamiento de los modelos correspondientes. Todas las arquitecturas consideradas son redes 2NN entrenadas en el subconjunto MNIST[4000/1000], con 10 % de ruido añadido y durante 5000 épocas. Se puede observar que, en términos globales, a mayor número de unidades (eje X) y mayor número de épocas (eje Y), mejores resultados a nivel de error en test (colores más oscuros). Más concretamente, hay una franja entre 5 y  $10^2$  épocas en donde parece que se acumulan los mejores resultados del primer descenso.

La Figura 7.16 muestra de manera clara ambos tipos de doble descenso. Por otra parte, en la imagen de la derecha se muestra el umbral de interpolación (punto en el que el modelo obtiene un error de entrenamiento cercano a 0), donde podemos observar como el cociente épocas/parámetros sigue una tendencia decreciente hasta estabilizarse, algo similar a lo que ocurre con el ratio parámetros/ejemplos (véase Subsección 7.2.4.1). Este resultado sugiere que, a medida que el tamaño del modelo aumenta, el número de épocas necesarias para alcanzar el umbral de interpolación se reduce de forma progresiva, hasta alcanzar un régimen en el que este cociente se mantiene prácticamente constante. Dicho comportamiento refuerza la idea de que la interpolación se vuelve más eficiente en términos de entrenamiento cuando la capacidad del modelo es suficientemente grande.

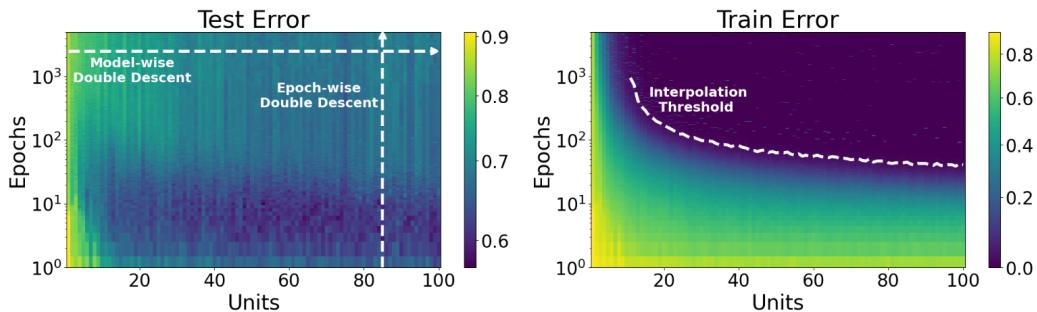


Figura 7.17.: A la izquierda, se muestra el error de test en función del tamaño del modelo y del número de épocas de entrenamiento. A la derecha, se representa el error de entrenamiento de los modelos correspondientes. Todas las arquitecturas consideradas son redes 2NN entrenadas en el subconjunto CIFAR10[4000/1000], con 20 % de ruido añadido, durante 5000 épocas y un *batch size* de 128. Los resultados obtenidos en este experimento son peores a los del conjunto de datos anterior, lo cual puede atribuirse a que la arquitectura presenta una complejidad insuficiente, lo que limita su capacidad de generalización.

Por otro lado, la Figura 7.17 no muestra, al menos de forma tan evidente, el doble descenso en el conjunto de datos CIFAR10. Esto sugiere que, aunque se ha utilizado la misma cantidad de datos, la mayor complejidad de este conjunto (al tratarse de imágenes en formato RGB) implica que la arquitectura 2NN no posee la capacidad suficiente para manifestar dicho comportamiento. Debido a este resultado, se decide no realizar el experimento con el conjunto de datos CIFAR100, ya que se anticipa un resultado igualmente desfavorable, dado que ambos conjuntos comparten el mismo formato de imágenes (véase Tabla 7.1).

Finalmente, en la Tabla 7.11 se recogen los tiempos de entrenamiento requeridos para la realización de este experimento.

Modelo	Dataset	Entrenamiento
2NN (1 – 100)	MNIST[4000/1000] (10 % label noise)	179h 5min
2NN (1 – 100)	CIFAR10[4000/1000] (20 % label noise)	213h 32min

Tabla 7.11.: Resumen de los experimentos para el doble descenso por complejidad del modelo y épocas para la red 2NN.

## 7. Análisis Empírico del Deep Double Descent

### 7.2.5.2. 3CNN

Continuamos con los experimentos realizados para la arquitectura 3CNN, la cual es más potente que la arquitectura 2NN debido a su mayor número de parámetros y a la inclusión de bloques convolucionales, que facilitan la extracción de características de las imágenes. Es por esto que, para la realización de estos experimentos, se considerarán subconjuntos de datos con un mayor número de ejemplos que en el apartado anterior.

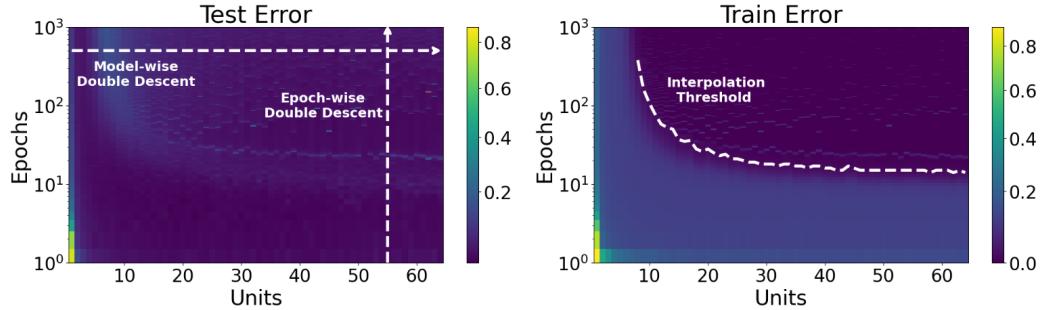


Figura 7.18.: A la izquierda, se muestra el error de test en función del tamaño del modelo y del número de épocas de entrenamiento. A la derecha, se representa el error de entrenamiento de los modelos correspondientes. Todas las arquitecturas consideradas son redes 3CNN entrenadas en el subconjunto MNIST[30000/5000], con 10 % de ruido añadido y un *batch size* de 128. Se puede observar una franja, representada en tonos azules, que indica un descenso en el rendimiento en los resultados de test y que sigue el patrón del umbral de interpolación de la gráfica de entrenamiento. Tras esta franja, el rendimiento mejora nuevamente, como se refleja en los colores más oscuros del gráfico.

La Figura 7.18 muestra con claridad ambos tipos de doble descenso sobre un subconjunto del conjunto de datos MNIST. Asimismo, se observa que el error obtenido para esta arquitectura es menor que el de la arquitectura anterior, lo que es coherente con su mayor capacidad.

Por otra parte, las Figuras 7.19 y 7.20 también muestran ambos tipos de doble descenso, aunque de forma menos evidente y con variaciones abruptas tanto en el error de entrenamiento como en el de test sobre dos subconjuntos de datos de CIFAR10 y CIFAR100. Estos saltos se analizan con más detalle en el Apéndice D. Asimismo, el error obtenido es mayor, dado que trabajamos con imágenes en formato RGB y de mayor dimensionalidad. Finalmente, la Tabla 7.12 recoge los tiempos de entrenamiento requeridos para la realización de este experimento.

Modelo	Dataset	Entrenamiento
3CNN (1 – 64)	MNIST[30000/5000] (10 % label noise)	179h 5min
3CNN (1 – 64)	CIFAR10[25000/5000] (20 % label noise)	147h 11min
3CNN (1 – 64)	CIFAR100[25000/5000] (20 % label noise)	152h 56min

Tabla 7.12.: Resumen de los experimentos para el doble descenso por complejidad del modelo y épocas para la arquitectura 3CNN.

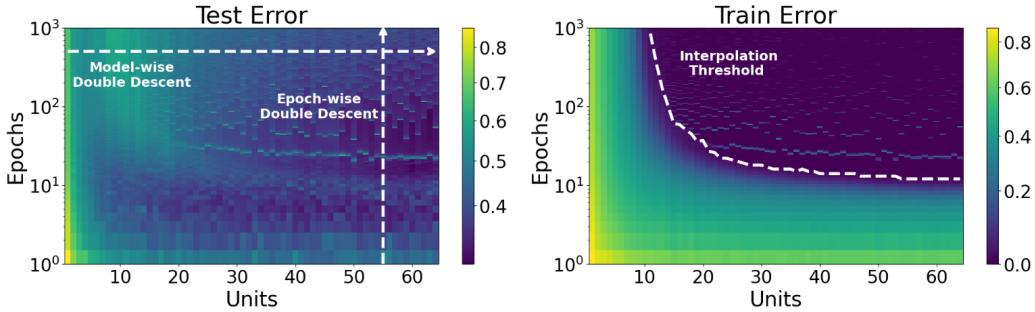


Figura 7.19.: A la izquierda, se muestra el error de test en función del tamaño del modelo y del número de épocas de entrenamiento; a la derecha, el error de entrenamiento correspondiente. Todas las arquitecturas son redes 3CNN entrenadas en CIFAR10[25000/5000], con 20 % de ruido añadido y un *batch size* de 128. En ambas gráficas se identifican regiones con repuntes de forma transitoria del error, evidenciados por la alternancia entre colores oscuros y tonos azules.

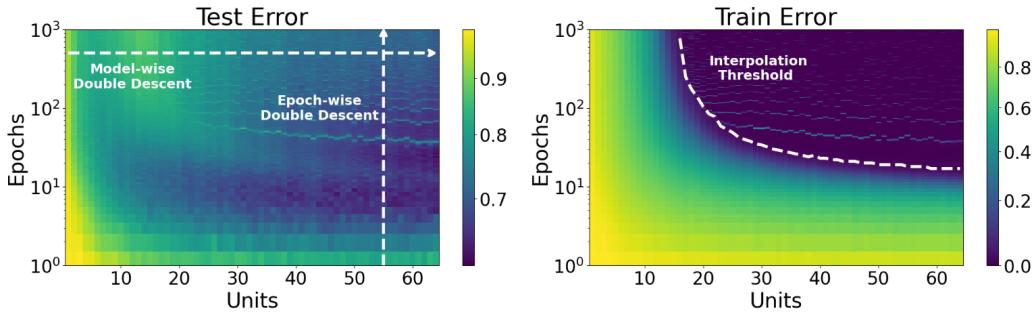


Figura 7.20.: A la izquierda, se muestra el error de test en función del tamaño del modelo y del número de épocas de entrenamiento; a la derecha, el error de entrenamiento correspondiente. Todas las arquitecturas son redes 3CNN entrenadas en CIFAR100[25000 / 5000], con 20 % de ruido añadido y un *batch size* de 128. Nuevamente se identifican variaciones transitorias del error y, además, se evidencia la mayor dificultad de este *dataset*, dado que los errores asociados son mayores.

### 7.2.5.3. ResNet18 modificada

Finalmente, se presentan los experimentos realizados con la arquitectura ResNet18 modificada, la cual posee el mayor número de parámetros entre las arquitecturas consideradas en este estudio. Dada su mayor capacidad de representación, se utilizan los conjuntos de datos en su totalidad para aprovechar al máximo su potencial.

Sin embargo, estos experimentos se vieron condicionados por la limitación impuesta por la Universidad de Granada en cuanto al tiempo máximo de uso de los nodos de cómputo. Como consecuencia, el número de épocas de entrenamiento se redujo de forma significativa, lo que ha afectado de manera significativa a la precisión de los resultados obtenidos.

Las Figuras 7.21, 7.22 y 7.23 muestran ambos tipos de doble descenso. No obstante, cabe destacar que el *epoch-wise double descent* no es tan notorio debido a la baja cantidad de épocas utilizadas, como consecuencia de la restricción impuesta por la propia UGR.

Cabe destacar que, para esta arquitectura, no se observan los saltos bruscos al incrementar

## 7. Análisis Empírico del Deep Double Descent

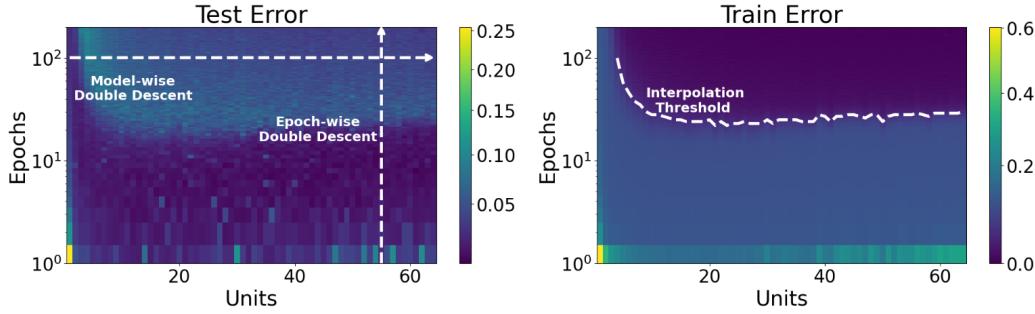


Figura 7.21.: A la izquierda, se muestra el error en test en función del tamaño del modelo y del número de épocas de entrenamiento. A la derecha, se representa el error de entrenamiento de los modelos correspondientes. Todas las arquitecturas consideradas son redes ResNet18 entrenadas en el conjunto MNIST, con 10 % de ruido añadido y durante 200 épocas. Se observa que el error de test comienza a mejorar nuevamente después de alcanzar las 100 épocas (colores más oscuros), aunque requiere un mayor número de épocas para volver a acercarse a los valores obtenidos durante el primer descenso.

el número de épocas en ninguno de los conjuntos de datos utilizados. Esto puede observarse con más detalle en la Figura 7.24, donde se presentan dos variantes modificadas de la arquitectura ResNet18 operando en el régimen sobreparametrizado, donde se evidencia de forma clara el *epoch-wise double descent*. Este comportamiento se analiza con mayor detalle en el Apéndice D. No obstante, de forma resumida, se atribuye a la presencia de conexiones residuales en la arquitectura, las cuales suavizan el paisaje de la función de pérdida, evitando que sea demasiado caótico (multimodal) y y facilitando la convergencia suave del modelo.

Finalmente, en la Tabla 7.13 se recogen los tiempos de entrenamiento requeridos para la realización de este experimento.

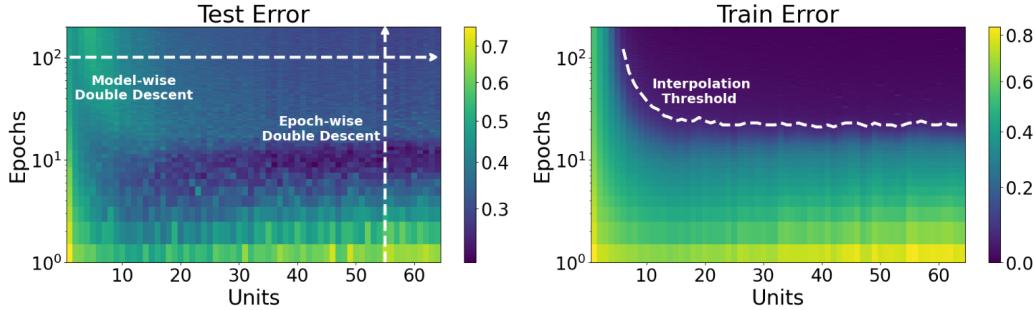


Figura 7.22.: A la izquierda, se muestra el error de test en función del tamaño del modelo y del número de épocas de entrenamiento. A la derecha, se representa el error de entrenamiento de los modelos correspondientes. Todas las arquitecturas consideradas son redes ResNet18 entrenadas en el conjunto CIFAR10 con aumento de datos<sup>8</sup>, 20 % de ruido añadido y durante 200 épocas. Se observa que el error de test mejora después de las 100 épocas, y se evidencia la mayor dificultad de este *dataset* al presentar errores mayores. También se aprecia cómo entre 3 y 10 épocas se concentran los mejores resultados tras el primer descenso.

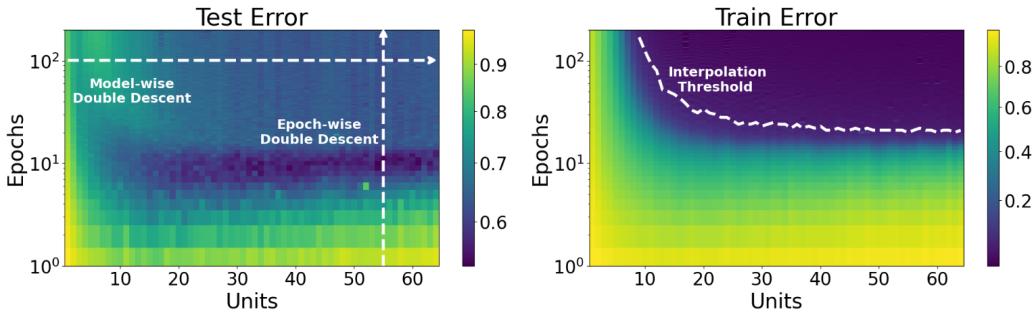


Figura 7.23.: A la izquierda, se muestra el error de test en función del tamaño del modelo y del número de épocas de entrenamiento. A la derecha, se representa el error de entrenamiento de los modelos correspondientes. Todas las arquitecturas consideradas son redes ResNet18 entrenadas en el conjunto CIFAR100 con aumento de datos, 20 % de ruido añadido y durante 200 épocas. Se observa nuevamente que el error de test mejora después de las 100 épocas. Asimismo, se evidencia la mayor dificultad de este *dataset* al presentar errores mayores que en el caso anterior, debido a que cuenta con 100 clases, lo que implica un mayor esfuerzo computacional para apreciar con precisión el DDD. Finalmente, se aprecia cómo entre las 5 y 10 épocas se concentran los mejores resultados.

Modelo	Dataset	Entrenamiento
PreActResNet18 (1 – 64)	MNIST (10 % label noise)	68h 33min
PreActResNet18 (1 – 64)	CIFAR10 (20 % label noise)	86h 17min
PreActResNet18 (1 – 64)	CIFAR100 (20 % label noise)	88h 51min
PreActResNet18 (45)	CIFAR10 (20 % label noise)	21h 26min
PreActResNet18 (64)	CIFAR10 (20 % label noise)	23h

Tabla 7.13.: Resumen de los experimentos para el doble descenso por complejidad del modelo y épocas para la red ResNet18 modificada.

#### 7.2.5.4. Discusión

Para concluir, se exponen las conclusiones más significativas derivadas de este experimento:

- Se han identificado numerosos casos favorables del fenómeno de doble descenso al utilizar diversas arquitecturas y conjuntos de datos. Además, se han observado casos desfavorables relacionados con la falta de capacidad de la arquitectura empleada para alcanzar su complejidad efectiva.
- El promedio entre épocas y parámetros en el que aparece el umbral de interpolación sigue una tendencia descendente hasta estabilizarse.
- Al entrenar modelos sobreparametrizados durante un gran número de épocas sin conexiones residuales, se observan saltos bruscos en los errores de entrenamiento y test, lo que está relacionado con la caoticidad del paisaje de la función de pérdida. No obstante, este paisaje resulta menos caótico cuando se emplean conexiones residuales.

<sup>8</sup>A partir de este momento, siempre que se hace referencia a “aumento de datos” implicará el uso conjunto de *RandomCrop(32, 4)* y *RandomHorizontalFlip* como técnicas de transformación sobre el conjunto de entrenamiento.

## 7. Análisis Empírico del Deep Double Descent

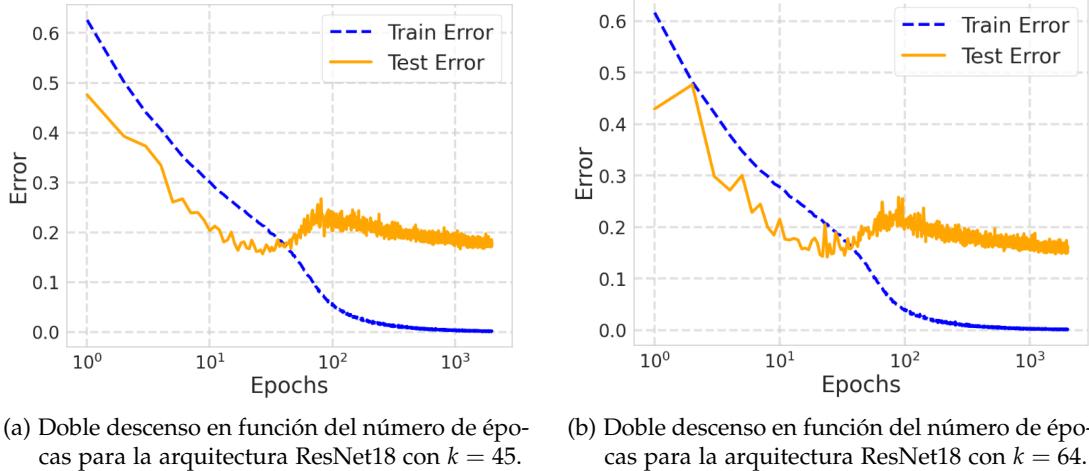


Figura 7.24.: Doble descenso en función del número de épocas para dos arquitecturas ResNet18 sobreparametrizadas, entrenadas sobre el conjunto CIFAR10 con aumento de datos y 20 % de ruido añadido, durante 2000 épocas.

### 7.2.6. Width vs Depth double descent

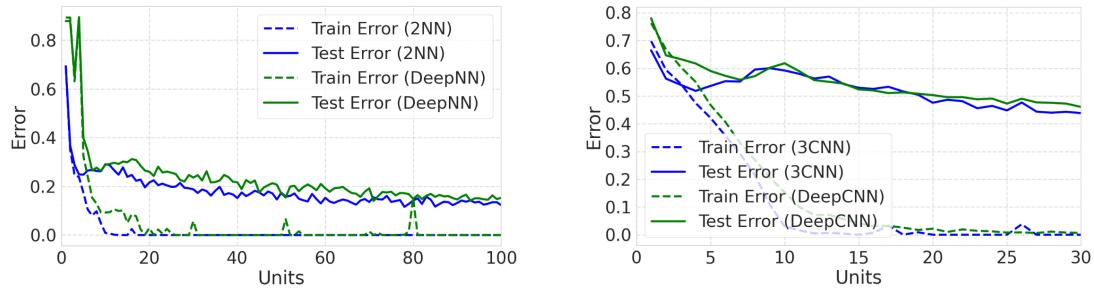
En esta subsección se analiza cómo varía el *Deep Double Descent* al emplear arquitecturas con la misma capacidad, pero con configuraciones diferentes. En particular, se comparan redes poco profundas y muy anchas con redes más profundas y poco anchas, dado que, según la teoría, ambas presentan comportamientos distintos ante un mismo problema [NRK21].

En la práctica, la literatura científica tiende a centrarse en redes anchas, cuya capacidad se incrementa aumentando el número de filtros en las capas convolucionales o el número de unidades ocultas en las capas densas. En contraste, existen pocos trabajos que exploren arquitecturas significativamente más profundas con la misma capacidad.

Para la realización de este experimento, se utilizan las arquitecturas 2NN y 3CNN como redes anchas y poco profundas. A partir de ellas, se incrementa la profundidad añadiendo capas densas a la 2NN y capas convolucionales a la 3CNN, y se reduce su anchura disminuyendo el número de neuronas y filtros por capa, manteniendo un número de parámetros similar. De esta forma, si consideramos que la red 2NN cuenta con 2 capas de profundidad (2 capas densas), la nueva red análoga profunda, DeepNN, cuenta con 8 capas de profundidad. A su vez, la red 3CNN cuenta con 3 capas de profundidad (3 capas convolucionales, sin contar la capa densa de salida), y su red análoga profunda, DeepCNN, cuenta con 10 capas de profundidad.

La Figura 7.25 muestra una comparación entre las arquitecturas análogas mencionadas en distintos conjuntos de datos. Se observa que el doble descenso se manifiesta de forma similar al aumentar la capacidad de una arquitectura, ya sea mediante mayor anchura o mayor profundidad. Asimismo, se observa que el umbral de interpolación se desplaza hacia la derecha al utilizar una red de mayor profundidad, ligado al hecho de que este tipo de arquitectura requiere más capacidad para alcanzar su complejidad efectiva.

Como conclusión, aunque la experimentación sea limitada, estos resultados sugieren que el fenómeno podría ser, en cierta medida, **robusto frente a la configuración utilizada**, lo que indica que no es necesario crear arquitecturas tan profundas para observar su aparición,



(a) Error de entrenamiento (línea discontinua) y test (línea continua) para dos arquitecturas (DeepNN y 2NN) con similar número de parámetros pero distinta configuración sobre el subconjunto MNIST[4000/1000] con 10 % de ruido añadido.

(b) Error de entrenamiento (línea discontinua) y test (línea continua) para dos arquitecturas (DeepCNN y 3CNN) con similar número de parámetros pero distinta configuración sobre el subconjunto CIFAR10[25000/5000] con 20 % de ruido añadido y un *batch size* de 128.

Figura 7.25.: Comparativa del doble descenso entre arquitecturas anchas y profundas. Se observa el DDD en ambas configuraciones, y se aprecia cómo, al utilizar redes más profundas, el *interpolation threshold* se desplaza hacia la derecha, necesitando la red más capacidad para alcanzar un error de entrenamiento cercano a 0.

trabajando así con redes más compactas.

#### 7.2.6.1. Discusión

Como resumen final, se presentan las conclusiones más relevantes derivadas de este experimento:

- Al emplear redes profundas en lugar de redes anchas, se sigue manifestando el DDD, lo que sugiere que este es robusto frente a diferentes configuraciones dentro de una misma arquitectura.
- El umbral de interpolación se desplaza hacia la derecha al utilizar redes profundas, lo que está relacionado con la necesidad del modelo de una mayor capacidad para alcanzar su complejidad efectiva.

### 7.3. Discusión comparativa global

A lo largo de esta sección se han presentado diversos experimentos destinados a analizar la aparición del doble descenso en distintas arquitecturas y conjuntos de datos. Asimismo, se han explorado casos en los que no se manifiesta.

El análisis comenzó con un ejemplo sencillo de regresión sobre datos sintéticos, sin emplear redes neuronales, utilizando la pseudoinversa para encontrar la solución. Este estudio puso de manifiesto la influencia de la base elegida, dado que al utilizar una base de polinomios de Legendre se observó el efecto, mientras que con las otras bases empleadas no se aprecia. También se evidenció cómo la presencia de ruido favorece su aparición y cómo las soluciones proporcionadas por la pseudoinversa favorecen aquellas con menor norma del vector de parámetros, tal como se explicó en la Sección 2.2. Asimismo, se verificó que el modelo tiende a preferir soluciones más simples (suaves), en concordancia con los principios teóricos.

## 7. Análisis Empírico del Deep Double Descent

Posteriormente, se introdujo el concepto de *noise-wise double descent*, destacando cómo el porcentaje de ruido incide tanto en la localización del máximo del error de generalización como en su magnitud. Se observó que, al aumentar el nivel de ruido, el máximo del error se desplaza hacia la derecha y se vuelve más pronunciado.

En el siguiente experimento, correspondiente al *sample-wise double descent*, se identificó una región en la que incrementar la cantidad de datos de entrenamiento puede, de manera paradójica, empeorar el rendimiento. A su vez, se vuelve a observar un desplazamiento del máximo del error de generalización conforme se incrementa el número de ejemplos, y se estimó un valor promedio de la razón entre parámetros y ejemplos en el que dicho máximo suele situarse.

Finalmente, se abordaron los dos casos más representativos: el *model-wise* y el *epoch-wise double descent*, tratados de manera conjunta. En esta sección se incluyeron resultados obtenidos con múltiples arquitecturas y conjuntos de datos, lo que permitió estudiar en qué condiciones se reproduce el fenómeno. También se examinó si el comportamiento se presenta por igual en redes anchas y profundas.

Para concluir, se presenta la Tabla 7.14 como resumen de los experimentos realizados, junto con una indicación sobre la presencia o ausencia del comportamiento en cada caso.

		Double - Descent				
Conjunto de datos	Arquitectura	% Ruido	Model	Epoch	Sample	Figura(s)
Sintético	Legendre base	0	✓	-	-	7.5, 7.11
	Polynomial base	0	✗	-	-	7.7a
	Redundant base	0	✗	-	-	7.7b
MNIST	2NN	0	✗	✗	-	7.12
		10	✓	✓	✓	7.14, 7.15
		[20 – 40]	✓	✓	-	7.12
	DeepNN	10	✓	-	-	7.25
CIFAR10	3CNN	10	✓	✓	-	7.18
	PreActResNet18	10	✓	✓	-	7.21
	2NN	20	✗	✗	-	7.17
	3CNN	20	✓	✓	-	7.19
CIFAR100	DeepCNN	20	✓	-	-	7.25
	PreActResNet18	20	✓	✓	-	7.22
	3CNN	20	✓	✗	-	7.20
	PreActResNet18	20	✓	✗	-	7.23

Tabla 7.14.: Resumen de los experimentos realizados para los distintos tipos de *Deep Double Descent* en función del conjunto de datos, la arquitectura y el porcentaje de ruido. El símbolo “–” indica que no se ha realizado el experimento correspondiente.

## **Parte IV.**

### **Conclusiones y Trabajos Futuros**



## 8. Conclusiones

La base teórica que sustenta los avances empíricos constituye una herramienta fundamental para el devenir del aprendizaje automático, y su desarrollo resulta esencial para comprender los nuevos comportamientos que van surgiendo en este ámbito. En el presente TFG, nos enfocamos en dar explicación al *Deep Double Descent* a partir del desarrollo de nuevos conceptos dentro del aprendizaje, unificando la sabiduría clásica con la experimentación moderna.

En primer lugar, se realizó un análisis exhaustivo del estado del arte relacionado con las distintas manifestaciones del fenómeno, constatando su novedosa y creciente relevancia en la comunidad científica, así como identificando sus principales desencadenantes tanto en problemas de regresión como de clasificación.

A continuación, nos centramos en revisar conceptos básicos de probabilidad y álgebra lineal, especialmente aquellos relativos a matrices, necesarios para introducir los fundamentos clásicos del aprendizaje. A su vez, estos conocimientos constituyen la base teórica sobre la que también se construyen y formalizan gran parte de los conceptos del aprendizaje moderno.

Posteriormente, nos adentramos en el estudio clásico del *deep learning*, presentando los principales problemas que este paradigma busca resolver y centrándonos especialmente en tareas de regresión y procesamiento de imágenes mediante redes neuronales convolucionales. Esta etapa sirve como preámbulo para abordar el dilema clásico del aprendizaje, permitiendo establecer un marco coherente que conecta los fundamentos estadísticos con los desafíos inherentes en este campo.

Una vez definido el concepto de aprendizaje, nos centramos en desarrollar los principales conceptos clásicos asociados, los cuales constituyen su base teórica fundamental. Esta revisión permitió establecer de forma precisa el marco teórico tradicional, sirviendo de punto de partida para comprender las diferencias y desafíos que surgen en el marco del aprendizaje moderno. De igual manera, nos permitió profundizar en la comprensión de conceptos clásicos de generalización, revelando metodologías de utilidad para el estudio de nuestro problema.

El núcleo del proyecto se centra en la exposición del *Deep Double Descent* desde una perspectiva tanto teórica como empírica. Inicialmente, se establece un planteamiento teórico del mismo, resolviendo sus discrepancias con la teoría clásica y realizando un análisis intuitivo de su aparición en problemas de regresión. Posteriormente, se presentan resultados que corroboran su aparición al emplear el descenso de gradiente como método de optimización.

Por otro lado, se realiza un estudio exhaustivo en la zona de sobreparametrización, donde se obtienen resultados que permiten aclarar su aparición desde un punto de vista teórico, a partir de la definición de sesgos inductivos que guían al modelo hacia soluciones efectivas, fundamentadas en su simplicidad, y que amplían la comprensión más allá de lo que la teoría clásica puede proporcionar, ofreciendo nuevos límites de generalización.

Para finalizar la parte teórica, se introducen conceptos de la aproximación no lineal, estrechamente ligados al concepto del aprendizaje y al propio fenómeno, destacando el uso de distintas bases y diccionarios redundantes de funciones, lo que proporciona una nueva perspectiva para su interpretación.

Para finalizar el proyecto, se lleva a cabo la constatación experimental de los resultados teóricos expuestos. En primer lugar, se comprueba que, al abordar problemas de regresión, la

## 8. Conclusiones

elección de la base utilizada desempeña un papel fundamental en la calidad de las sucesivas aproximaciones. Por otro lado, en tareas de clasificación de imágenes, se observa cómo el fenómeno emerge en contextos típicos del aprendizaje automático, especialmente cuando el modelo supera de largo su capacidad efectiva, ya sea por exceso de parámetros o al realizar el entrenamiento durante un tiempo excesivo. Asimismo, se corrobora que el ruido desempeña un papel crucial tanto en su aparición como en su intensidad.

A modo de resumen de la parte experimental, se listan a continuación las principales conclusiones obtenidas acerca del DDD:

- En problemas de regresión, la base utilizada es clave para su aparición, destacando su manifestación al emplear la base de Legendre. En las aproximaciones sucesivas, esta base muestra un componente de suavizado o regularización implícita que hace que la aproximación final se asemeje a la inicial, siguiendo la filosofía de Ockham.
- El ruido también influye de forma significativa en su aparición, haciendo que se manifieste en escenarios donde, en ausencia de ruido, no lo hace. Además, tanto la magnitud como la posición del máximo del error de test dependen del nivel de ruido, siendo este más pronunciado y desplazado hacia la derecha a medida que el ruido aumenta.
- El número de ejemplos de entrenamiento afecta a la localización del umbral de interpolación. A medida que se incrementa la cantidad de datos, dicho umbral se desplaza hacia la derecha, lo que genera una zona en la que entrenar con más datos puede empeorar temporalmente el rendimiento del modelo.
- La localización del umbral de interpolación, en función del ratio entre parámetros y ejemplos, tiende a aproximarse y estabilizarse en torno a 1.
- La estructura de la red no parece tener un impacto determinante en su aparición, ya que este se manifiesta tanto en redes anchas como en redes profundas que presentan una arquitectura similar. No obstante, en redes más profundas, el umbral de interpolación se desplaza hacia la derecha.
- En arquitecturas sin conexiones residuales se observan repuntes transitorios del error de entrenamiento y test, asociados a la caoticidad del paisaje de la función de pérdida.

Por ende, el presente trabajo ha establecido una base sólida para comprender el *Deep Double Descent*. Además, actúa como un puente entre los enfoques clásicos y modernos, ofreciendo una primera aproximación hacia el mundo de la generalización en escenarios de sobreparametrización. No obstante, aún existe un amplio margen para descubrir principios subyacentes a este comportamiento y explorar sus posibles manifestaciones y su relevancia en diversos problemas del mundo actual.

Para el desarrollo de este TFG se han aplicado conocimientos adquiridos en asignaturas como Aprendizaje Automático, Aprendizaje Profundo, Visión por Computador, Probabilidad y Geometría. Además, se han conseguido nuevos conocimientos relacionados con la Aproximación No Lineal, conceptos clásicos y nuevas teorías del Aprendizaje Profundo, así como el uso práctico de herramientas y métodos basados en la biblioteca *PyTorch*.

A modo de conclusión, podemos asegurar que los objetivos presentados al inicio del trabajo se han cumplido de manera satisfactoria, cumpliendo con las expectativas establecidas. No obstante, como se detallará en la siguiente sección, han surgido diversas oportunidades para futuras investigaciones, impulsadas tanto por la naturaleza innovadora de este estudio como por las restricciones de tiempo y recursos computacionales.

## 9. Trabajos futuros

Aunque nuestro trabajo ha representado un avance significativo en la comprensión de nuevas dinámicas de actuación de las arquitecturas profundas, especialmente en términos de generalización, aún existen diversos aspectos que requieren un análisis más profundo. Ligado a esto se suma el carácter emergente y en constante evolución de este campo dentro de la IA, que invita a seguir explorando nuevas hipótesis y enfoques que permitan construir un conocimiento más sólido de estos comportamientos.

En primer lugar, una línea de trabajo evidente sería profundizar en experimentos más ambiciosos que no hemos podido llevar a cabo debido a limitaciones en la capacidad de cómputo. Para ello, se propone emplear arquitecturas con una capacidad extremadamente mayor y entrenarlas durante un período de tiempo más prolongado, con el fin de corroborar que, en la mayoría de los casos, se alcanzarán errores inferiores a los del primer descenso. Más aún, sería interesante explorar el uso de otro tipo de arquitecturas de gran escala, tales como los *Transformers* o modelos basados en técnicas de *ensembling*.

Por otra parte, sería interesante indagar en la relación entre el *Deep Double Descent* y el *Groking*, dado que ambos parecen representar un mismo escenario en términos de generalización en modelos sobreparametrizados. De esta manera, se podría avanzar hacia la unificación de las nuevas tendencias emergentes en el mundo de la generalización dentro de un marco teórico común.

Otra línea de investigación con gran potencial consistiría en llevar a cabo un análisis preliminar del problema a abordar, con el objetivo de estimar de forma anticipada el tiempo y la capacidad computacional necesarios para alcanzar mejoras en el rendimiento durante el segundo descenso. De este modo, se buscaría desarrollar una herramienta capaz de predecir el coste estimado necesario para producir mejoras de rendimiento con respecto a las del mínimo alcanzado durante el descenso clásico inicial.

Volviendo a los experimentos, sería razonable estudiar el comportamiento del fenómeno ante datos que incluyan distintos tipos de ruido, no solo en las etiquetas, sino también en las propia entrada. Esto permitiría observar cómo se comporta frente a distintos escenarios que pueden presentarse en el mundo real, dado que el ruido en estos entornos no sigue, por lo general, una distribución uniforme.

Un campo adicional de exploración consistiría en abordar problemas de regresión que utilicen modelos neuronales, con el objetivo de verificar si existen matices o diferencias en comparación con los problemas de clasificación. Además, en estos escenarios, sería interesante explorar un mayor número de bases para la aproximación y estudiar las razones por las cuales ciertas bases pueden presentar el doble descenso mientras que otras no lo hacen.

Como se ha mencionado anteriormente, existe un amplio margen de mejora, especialmente en el ámbito empírico. Sin embargo, el estudio del *Deep Double Descent* ha arrojado luz sobre la explicabilidad de la generalización en la actualidad. Este hecho pone de manifiesto la importancia de no ignorar estas nuevas tendencias, sino de trabajar para conseguir una integración con los enfoques tradicionales, con el objetivo de sentar las bases que permitan unificar los avances recientes con los fundamentos clásicos.

## **Parte V.**

### **Anexos**



## A. Detalles Matemáticos Adicionales

En este apéndice se muestran las demostraciones del Lema 6.7, extraída de [SHN<sup>+</sup>24], y de la Proposición 6.9.

**Lema 6.7.** Sea  $\mathcal{L}(w)$  una función  $\beta$ -suave no negativa. Si  $\eta < \frac{2}{\beta}$ , entonces, para cualquier  $w_0$  y utilizando el método del descenso de gradiente dado por:

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla \mathcal{L}(w),$$

se tiene que  $\sum_{u=0}^{\infty} \|\nabla \mathcal{L}(w^{(u)})\|^2 < \infty$  y, por tanto:

$$\lim_{t \rightarrow \infty} \|\nabla \mathcal{L}(w^{(t)})\|^2 = 0.$$

*Demostración.* Usaremos una propiedad conocida de las funciones  $\beta$ -suaves:

$$|f(x) - f(y) - \nabla f(y)^T(x - y)| \leq \|x - y\|^2.$$

Dado que la función  $\mathcal{L}(w)$  es  $\beta$ -suave:

$$\begin{aligned} \mathcal{L}(w^{(\tau+1)}) &\leq \mathcal{L}(w^{(\tau)}) + \nabla \mathcal{L}(w^{(\tau)})^T(w^{(\tau+1)} - w^{(\tau)}) + \frac{\beta}{2} \|w^{(\tau+1)} - w^{(\tau)}\|^2 \\ &= \mathcal{L}(w^{(\tau)}) - \eta \|\nabla \mathcal{L}(w^{(\tau)})\|^2 + \frac{\beta \eta^2}{2} \|\nabla \mathcal{L}(w^{(\tau)})\|^2 \\ &= \mathcal{L}(w^{(\tau)}) - \eta \left(1 - \frac{\beta \eta}{2}\right) \|\nabla \mathcal{L}(w^{(\tau)})\|^2. \end{aligned}$$

Así, tenemos:

$$\frac{\mathcal{L}(w^{(\tau)}) - \mathcal{L}(w^{(\tau+1)})}{\eta \left(1 - \frac{\beta \eta}{2}\right)} \geq \|\nabla \mathcal{L}(w^{(\tau)})\|^2,$$

lo que implica que

$$\sum_{u=0}^t \|\nabla \mathcal{L}(w^{(u)})\|^2 \leq \sum_{u=0}^t \frac{\mathcal{L}(w^{(u)}) - \mathcal{L}(w^{(u+1)})}{\eta \left(1 - \frac{\beta \eta}{2}\right)} = \frac{\mathcal{L}(w_0) - \mathcal{L}(w^{(\tau+1)})}{\eta \left(1 - \frac{\beta \eta}{2}\right)}.$$

El lado derecho está acotado por una constante finita, dado que  $\mathcal{L}(w_0) < \infty$  y  $0 \leq \mathcal{L}(w^{(\tau+1)})$ . Por tanto, se deduce que

$$\sum_{u=0}^{\infty} \|\nabla \mathcal{L}(w^{(u)})\|^2 < \infty,$$

y, finalmente, nos queda el resultado buscado:  $\|\nabla \mathcal{L}(w^{(\tau)})\|^2 \rightarrow 0$ . □

**Proposición 6.9.** Sea  $w^*$  una solución, es decir,  $\mathcal{L}(w^*) = 0$ , y supongamos que  $\frac{d}{dw} \frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \neq 0$  y  $\text{rang}(H_{\mathcal{F}_i}(w^*)) > 2n$  para algún  $i \in \{1, \dots, n\}$ . Entonces,  $\mathcal{L}(w)$  no es convexa en ningún entorno de  $w^*$ .

*Demostración.* La matriz Hessiana de una función de pérdida  $\mathcal{L}(\mathcal{F}(w))$  viene dada por:

$$H_{\mathcal{L}}(w) = \mathcal{D}\mathcal{F}(w)^T \frac{\partial^2 \mathcal{L}}{\partial \mathcal{F}^2}(w) \mathcal{D}\mathcal{F}(w) + \sum_{i=1}^n \frac{\partial \mathcal{L}}{\partial \mathcal{F}_i}(w) H_{\mathcal{F}_i}(w).$$

Consideramos ahora las matrices Hessianas de dos puntos  $w^* + \delta$  y  $w^* - \delta$  ( $\delta \in \mathbb{R}^P$ ) que están en un entorno suficientemente pequeño de  $w^*$ . La Hessiana de la función de pérdida en estos dos puntos es:

$$\begin{aligned} H_{\mathcal{L}}(w^* + \delta) &= \underbrace{\mathcal{D}\mathcal{F}(w^* + \delta)^T \frac{\partial^2 \mathcal{L}}{\partial \mathcal{F}^2}(w^* + \delta) \mathcal{D}\mathcal{F}(w^* + \delta)}_{A(w^* + \delta)} \\ &\quad + \sum_{i=1}^n \left( \frac{d}{dw} \left( \frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) \delta \right)_i H_{\mathcal{F}_i}(w^* + \delta) + o(\|\delta\|), \\ H_{\mathcal{L}}(w^* - \delta) &= \underbrace{\mathcal{D}\mathcal{F}(w^* - \delta)^T \frac{\partial^2 \mathcal{L}}{\partial \mathcal{F}^2}(w^* - \delta) \mathcal{D}\mathcal{F}(w^* - \delta)}_{A(w^* - \delta)} \\ &\quad - \sum_{i=1}^n \left( \frac{d}{dw} \left( \frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) \delta \right)_i H_{\mathcal{F}_i}(w^* - \delta) + o(\|\delta\|), \end{aligned}$$

donde los términos  $A(w^* + \delta)$  y  $A(w^* - \delta)$  son matrices de rango a lo sumo  $n$ , ya que  $\mathcal{D}\mathcal{F}$  es de tamaño  $n \times m$  ( $o(\|\delta\|)$  representa un término asintóticamente menor que  $\|\delta\|$ ).

Sabemos que al menos un componente  $H_{\mathcal{F}_k}$  de la matriz Hessiana de  $\mathcal{F}$  satisface que el rango de  $H_{\mathcal{F}_k}(w^*)$  es mayor que  $2n$ . Por continuidad de la Hessiana, si el orden de  $\delta$  es suficientemente pequeño, entonces los rangos de  $H_{\mathcal{F}_k}(w^* + \delta)$  y  $H_{\mathcal{F}_k}(w^* - \delta)$  también son mayores que  $2n$ . Esto nos lleva a que siempre podemos encontrar un vector unitario  $v \in \mathbb{R}^P$  que cumpla:

$$v^T A(w^* + \delta) v = v^T A(w^* - \delta) v = 0.$$

Sin embargo, dicho vector también verifica:

$$v^T H_{\mathcal{F}_k}(w^* + \delta) v \neq 0, \quad v^T H_{\mathcal{F}_k}(w^* - \delta) v \neq 0.$$

De esta manera, el vector  $\langle v^T H_{\mathcal{F}_1}(w^* + \delta) v, \dots, v^T H_{\mathcal{F}_n}(w^* + \delta) v \rangle \neq 0$ . El mismo resultado se verifica para el punto  $w^* - \delta$ . Así, utilizando este vector  $v$  en la Hessiana de la función de pérdida en los dos puntos tenemos:

$$v^T H_{\mathcal{L}}(w^* + \delta) v = \sum_{i=1}^n \left( \frac{d}{dw} \left( \frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) \delta \right)_i v^T H_{\mathcal{F}_i}(w^* + \delta) v + o(\|\delta\|),$$

### A. Detalles Matemáticos Adicionales

$$v^T H_{\mathcal{L}}(w^* - \delta)v = - \sum_{i=1}^n \left( \frac{d}{dw} \left( \frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) \delta \right)_i v^T H_{\mathcal{F}_i}(w^* - \delta)v + o(\|\delta\|).$$

Para finalizar tenemos que mostrar que, para un  $\delta$  suficientemente pequeño,  $v^T H_{\mathcal{L}}(w^* + \delta)v$  y  $v^T H_{\mathcal{L}}(w^* - \delta)v$  no pueden ser simultáneamente no negativos, lo que implica inmediatamente que  $H_{\mathcal{L}}$  no es semidefinida positiva en un entorno cercano de  $w^*$  y, por tanto,  $\mathcal{L}$  no es localmente convexa en  $w^*$ . Usando la condición

$$\frac{d}{dw} \left( \frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) = 0,$$

para las ecuaciones anteriores, se presentan los siguientes casos:

**Caso 1:** Si

$$\sum_{i=1}^n \left( \frac{d}{dw} \left( \frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) \delta \right)_i v^T H_{\mathcal{F}_i}(w^* + \delta)v < 0,$$

entonces, directamente,  $v^T H_{\mathcal{L}}(w^* + \delta)v < 0$  si  $\delta$  es lo suficientemente pequeño.

**Caso 2:** De lo contrario, si

$$\sum_{i=1}^n \left( \frac{d}{dw} \left( \frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) \delta \right)_i v^T H_{\mathcal{F}_i}(w^* + \delta)v > 0,$$

por la continuidad de cada  $H_{\mathcal{F}_i}(\cdot)$ , tenemos que

$$\begin{aligned} & - \sum_{i=1}^n \left( \frac{d}{dw} \left( \frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) \delta \right)_i v^T H_{\mathcal{F}_i}(w^* - \delta)v \\ &= - \sum_{i=1}^n \left( \frac{d}{dw} \left( \frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) \delta \right)_i v^T H_{\mathcal{F}_i}(w^* + \delta)v + \sum_{i=1}^n \left( \frac{d}{dw} \left( \frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) \delta \right)_i v^T (H_{\mathcal{F}_i}(w^* + \delta)v - H_{\mathcal{F}_i}(w^* - \delta))v \\ &= - \sum_{i=1}^n \left( \frac{d}{dw} \left( \frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) \delta \right)_i v^T H_{\mathcal{F}_i}(w^* + \delta)v + O(\epsilon) < 0, \end{aligned}$$

cuando  $\delta$  es lo suficientemente pequeño. A modo de conclusión, dado un  $\delta$  arbitrariamente pequeño, se tiene que  $v^T H_{\mathcal{L}}(w^* + \delta)v$  o  $v^T H_{\mathcal{L}}(w^* - \delta)v$  es negativo, lo que significa que  $\mathcal{L}(w)$  no es convexa en ningún entorno de  $w^*$ .

□

## B. Experimentos Adicionales: Hiperparámetros

En este apéndice se presentan dos experimentos relacionados con el uso de diferentes tamaños de *batch* y *learning rates* para un modelo que exhibe el DDD, con la finalidad de verificar su comportamiento ante distintas configuraciones. Para ello, se empleará la arquitectura 2NN sobre el subconjunto MNIST[4000/1000], al cual se le añadirá un 10 % de ruido, y sobre el que se probarán diversos tamaños de *batch* y *learning rates*.

### Double descent con distinto tamaño de batch

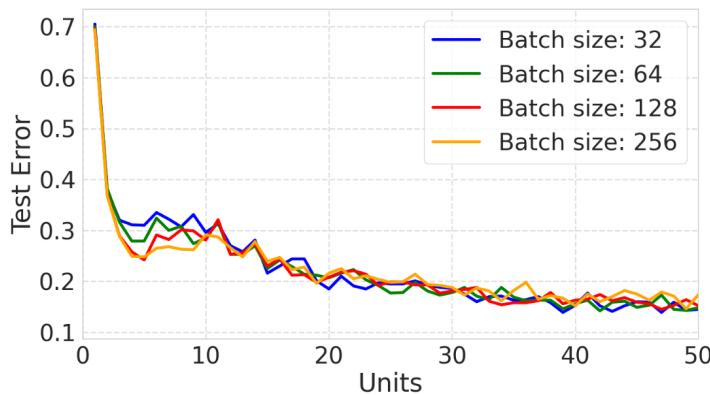


Figura B.1.: Error en test respecto a distintas configuraciones de tamaño de *batch* de la red 2NN sobre el subconjunto MNIST[4000/1000] con 10 % de ruido añadido. Se observa el fenómeno bajo las diferentes configuraciones, destacando que, al emplear tamaños de *batch* mayores, el error tiende a presentar menos oscilaciones.

En la Figura B.1 se observa que, para distintas configuraciones de tamaño de *batch*, se manifiesta el *Deep Double Descent*, siendo más notable (sin tantos altibajos en la curva del error) al utilizar tamaños mayores, debido a que se utiliza un mayor número de ejemplos para actualizar los parámetros en cada iteración. Por otro lado, la Tabla B.1 muestra que, al incrementar el tamaño de *batch*, el tiempo de entrenamiento disminuye, lo cual es lógico, ya que se procesan un mayor número de ejemplos en cada iteración. Por tanto, considerando ambos resultados, se concluye que utilizar un tamaño de *batch* elevado (128 o 256) es la opción adecuada para los distintos experimentos a realizar, ya que permite observar el fenómeno de forma más precisa y con menor coste computacional.

### Double descent con distinto learning rate

En la Figura B.2 podemos observar cómo se manifiesta el DDD para distintas configuraciones de *learning rate*. En particular, el pico del error de test se hace más pronunciado a medida que el *learning rate* aumenta, dado que para un valor igual a  $5 \times 10^{-4}$ , el máximo error en

## B. Experimentos Adicionales: Hiperparámetros

Modelo	Dataset	Batch size	Entrenamiento
2NN (1 – 50)	MNIST[4000/1000 – 10 % noise]	32	15h 2min
2NN (1 – 50)	MNIST[4000/1000 – 10 % noise]	64	11h 56min
2NN (1 – 50)	MNIST[4000/1000 – 10 % noise]	128	11h 14min
2NN (1 – 50)	MNIST[4000/1000 – 10 % noise]	256	10h 23min

Tabla B.1.: Resumen de los experimentos realizados para el *Deep Double Descent* con distinto tamaño de *batch* (*batch size*).



Figura B.2.: Error en test respecto a distintas configuraciones de *learning rate* obtenido por la red 2NN sobre el subconjunto MNIST[4000/1000] con 10 % de ruido añadido. Se observa que, al utilizar valores más elevados para el *learning rates*, el valor máximo del error es mayor.

test es mayor que para el valor por defecto del optimizador Adam ( $10^{-4}$ ), y este, a su vez, es mayor que el máximo valor que proporciona el mínimo *learning rate* utilizado ( $5 \times 10^{-5}$ ).

En conclusión, esto nos indica que, si un modelo presenta el *Deep Double Descent*, no es necesario preocuparse en exceso por ajustar de manera precisa el *learning rate*, puesto que, dentro de ciertos rangos de valores de este hiperparámetro, el comportamiento característico del fenómeno se mantiene, lo que indica que el valor por defecto del mismo resulta ser suficiente para observar y estudiar dicho comportamiento sin afectar significativamente los resultados, lo que contribuye a simplificar el proceso de diseño experimental.

Finalmente, para concluir esta subsección, se muestra en la Tabla B.2 un resumen de los experimentos realizados, junto con sus respectivos tiempos de entrenamiento. Cabe destacar que el uso de distintos *learning rates* no altera significativamente el tiempo de entrenamiento, dado que en todos los casos se realiza el mismo número de iteraciones.

Modelo	Dataset	Learning rate	Entrenamiento
2NN (1 – 100)	MNIST[4000/1000 – 10 % noise]	0.005	26h 23min
2NN (1 – 100)	MNIST[4000/1000 – 10 % noise]	0.001 ( <i>default</i> )	26h 50min
2NN (1 – 100)	MNIST[4000/1000 – 10 % noise]	0.0005	27h 2min

Tabla B.2.: Resumen de los experimentos realizados para el DDD con distinto *learning rate*.

## C. Descenso de Gradiente en un Problema OLS

En este anexo se investiga el comportamiento del descenso de gradiente al aplicarse a un problema OLS, en comparación con la solución obtenida mediante la pseudoinversa.

El objetivo es encontrar la solución al mismo problema de regresión descrito en el apartado 7.2.2.1, utilizando el descenso de gradiente y el error cuadrático medio como función de pérdida a minimizar, tanto para la base de Legendre como para la base polinomial clásica (omitiéndose la base redundante, al esperarse resultados similares a la base clásica).

Para los valores de los hiperparámetros, se utiliza una tasa de aprendizaje de  $10^{-2}$  y un número de épocas igual a 10000.

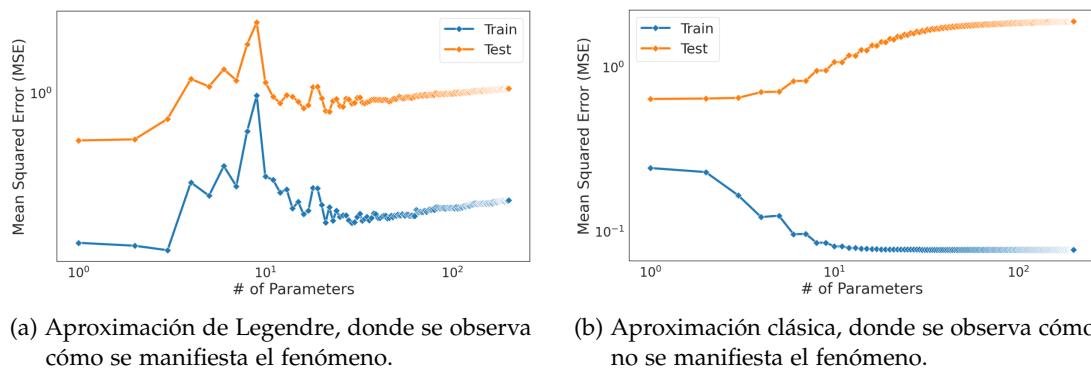


Figura C.1.: Error en entrenamiento y test para las bases de Legendre y clásica utilizando el GD. Se observa la aparición del *Deep Double Descent* al emplear la base de Legendre en ambos errores. En contraste, para la base polinomial clásica no se evidencia, aunque el error de test se estabiliza en un valor relativamente bajo.

La Figura C.1 muestra que los resultados obtenidos al utilizar el GD son similares a los obtenidos mediante la pseudoinversa (véase el apartado 7.2.2.1, donde se manifiesta el DDD para la base de Legendre y no para la base polinomial clásica), con la salvedad de que el error de entrenamiento es mayor al emplear GD y no llega a ser nulo, aunque es bastante cercano a cero. Esto se debe a que, en este caso, no estamos resolviendo de manera exacta una ecuación, sino que simplemente estamos aproximando la solución mediante un proceso iterativo, el cual requiere más tiempo para alcanzar la convergencia.

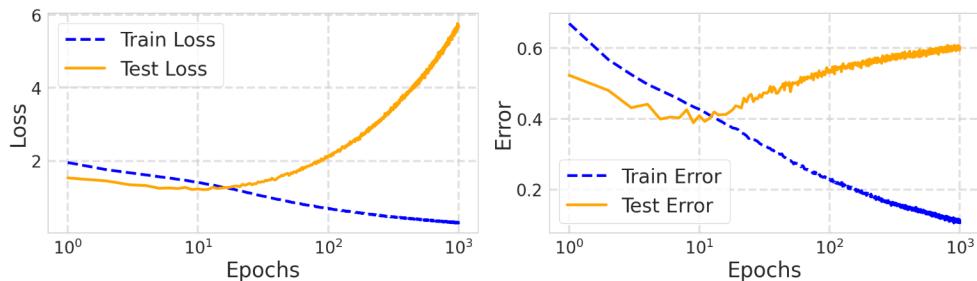
No obstante, ambas gráficas presentan errores de generación menores que los mostrados en el apartado 7.2.2.1, lo que sugiere que, antes de la convergencia total del GD hacia la solución dada por la pseudoinversa, el descenso de gradiente se queda con soluciones que pueden generalizar mejor que la propia solución de norma mínima para ese grado de complejidad.

Como conclusión, se evidencia que, al emplear un gran tamaño de *batch* y suficiente tiempo de entrenamiento (número de épocas), el descenso de gradiente tiende a aproximarse hacia la solución analítica que ofrece la pseudoinversa, tal y como se ha detallado (bajo ciertas condiciones) en la Subsección 6.2.1, y como también se expone en el trabajo de Lee et al. [LSJR16].

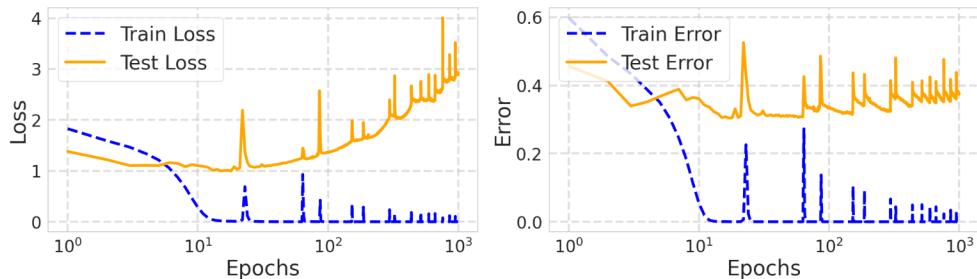
## D. Irregularidades en la Dinámica del Error

En esta sección complementaria analizamos en detalle los saltos bruscos que se producen tanto en el error de entrenamiento como en el de test al aumentar el número de épocas de un determinado modelo, intentando proporcionar una explicación para los mismos.

Nos centramos en la gráfica de la Figura 7.19, donde estas colinas en el error son más sobresalientes (no obstante, la Figura D.2 muestra un comportamiento similar al emplear un subconjunto de MNIST y la arquitectura 2NN, particularmente al utilizar modelos de gran capacidad). A partir de esta representación, se analiza el *epoch-wise double descent* en dos modelos que la conforman: uno infraparametrizado, ubicado antes del umbral de interpolación, y otro sobreparametrizado, situado después de dicho umbral (véase Figura D.1).



(a) A la izquierda, se muestra la pérdida en entrenamiento y test, y a la derecha, el error en entrenamiento y test para un modelo 3CNN infraparametrizado ( $k = 9$ ) sobre el subconjunto CIFAR10[25000/5000], con 20 % de ruido añadido y un batch size de 128.



(b) A la izquierda, se muestra la pérdida en entrenamiento y test, y a la derecha, el error en entrenamiento y test para un modelo 3CNN sobreparametrizado ( $k = 60$ ) sobre el subconjunto CIFAR10[25000/5000], con 20 % de ruido añadido y un batch size de 128. Se aprecian variaciones abruptas tanto en la gráfica de la pérdida como en la del error.

Figura D.1.: Comparativa *epoch-wise double descent* entre un modelo infraparametrizado, donde no se observa el DDD, y un modelo sobreparametrizado, en el que este comportamiento sí se manifiesta y aparecen colinas significativas en el error.

Para intentar dar una explicación coherente a la conducta, ciertamente peculiar, que se observa en estas gráficas, recordemos algunos de los resultados presentados en la Sección 6.3. En la zona infraparametrizada, sabemos que los mínimos locales suelen estar aislados, lo que

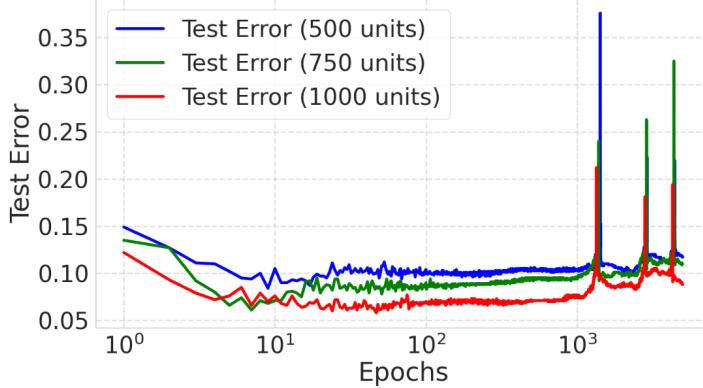


Figura D.2.: Error en test respecto a distintas configuraciones altamente sobreparametrizadas de la red 2NN sobre el subconjunto MNIST[4000/1000], con 10 % de ruido añadido y entrenadas durante 5000 épocas. Se observan repuntes en la curva del error, los cuales se atenúan progresivamente a medida que se avanza en el entrenamiento. Destacamos también que dichos picos tienden a ser menos pronunciados cuando el tamaño del modelo es mayor.

provoca que el paisaje de la función de pérdida en sus alrededores sea localmente convexo (véase imagen izquierda de la Figura 6.4).

Debido a esto, la inicialización del optimizador juega un papel crucial, ya que determinará hacia qué mínimo local convergerá. En particular, el optimizador tiende a dirigirse hacia el mínimo local más cercano a su punto de inicio. Dado que la región circundante es localmente convexa, el optimizador avanzará progresivamente hacia dicho mínimo sin posibilidad de escapar de él, siempre que la tasa de aprendizaje sea adecuada. Este comportamiento explica por qué, en la zona infraparametrizada, la pérdida en entrenamiento sigue una tendencia descendente y converge de manera estable. Al no haber direcciones de escape significativas, el optimizador se mantiene atrapado en el mínimo local alcanzado, lo que se traduce en una reducción progresiva y sostenida de la pérdida a lo largo del proceso de optimización. Sin embargo, debido a la limitada capacidad del modelo en esta región, el error en el conjunto de test permanece elevado, reflejando una falta de flexibilidad para ajustarse adecuadamente a los datos (véase Figura D.1a).

En cambio, en la zona sobreparametrizada, los mínimos globales no están aislados, sino que en cualquier pequeño entorno de ellos siempre es posible encontrar otro minimizador global (véase Sección 6.3). Esto implica que la función de pérdida presenta una alta no convexidad, incluso a nivel local. Como resultado, el paisaje de la función de pérdida en estos modelos es multimodal, como se ilustra en la imagen izquierda de la Figura D.3. Aunque dicha imagen muestre únicamente un mínimo global en un espacio tridimensional, es importante recordar que existen múltiples de ellos en las dimensiones definidas por el espacio de parámetros. En otras palabras, paisajes similares a este (aunque más complejos debido a su alta dimensionalidad) se replicarían en distintas regiones del espacio de parámetros, dando lugar a un paisaje altamente multimodal con una multitud de mínimos globales.

A partir de estos resultados, podemos inferir que el algoritmo de optimización, una vez ha alcanzado una solución prácticamente perfecta (es decir, al encontrar un mínimo, potencialmente global, de la función de pérdida), continúa explorando en busca de soluciones aún mejores (en términos de generalización). Este proceso, debido a la naturaleza del paisaje

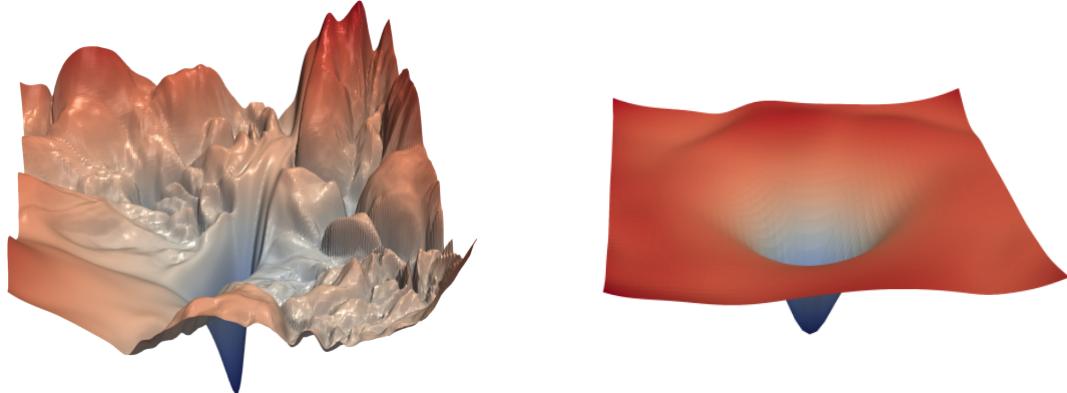


Figura D.3.: Paisajes de la función de pérdida para un modelo sobreparametrizado sin conexiones residuales (izquierda) y con conexiones residuales (derecha) [LXT<sup>+</sup>18]. A la izquierda, se observa un único mínimo global en un entorno altamente multimodal. A la derecha, se visualiza también un único mínimo global, pero en un entorno unimodal, fruto de las conexiones residuales.

multimodal en el que opera, provoca que el modelo pueda salir de dicho mínimo, lo que da lugar a los picos observados, para luego converger hacia otro mínimo potencialmente global.

Además, otro hecho relevante es que la magnitud de estos saltos disminuye progresivamente, lo que sugiere que el modelo, al buscar nuevos mínimos globales, tiende a seleccionar un tipo de soluciones frente a otras. Esta respuesta del modelo puede asociarse al hecho de que, entre todas las soluciones posibles, busca aquella que sea más simple, lo cual, como se ha observado en experimentos previos y en la parte teórica, suele ser una solución que generaliza bien.

La pregunta ahora sería por qué hemos observado estos resultados, cuando no se muestran de manera prominente en la literatura científica. En primer lugar, en los estudios que presentan gráficas de error para el *Deep Double Descent*, generalmente se utilizan arquitecturas que incluyen conexiones residuales. Este tipo de arquitecturas, según lo demuestran Li et al. [LXT<sup>+</sup>18], no presentan paisajes de la función de pérdida tan caóticos (multimodales). En cambio, exhiben zonas planas alrededor de los mínimos, impidiendo los saltos bruscos observados en nuestro caso (véase imagen derecha de la Figura D.3). Así, los mínimos globales de las arquitecturas con conexiones residuales se asemejan a los mínimos locales de las zonas infraparametrizadas, lo que impide que el algoritmo de optimización escape de ellos.

Por otra parte, siguiendo los distintos resultados presentados en la Sección 7.2.5, podemos observar que el aumento en el número de parámetros en comparación con el número de ejemplos de entrenamiento, junto con el uso de un conjunto de entrenamiento más difícil (en nuestro caso, conjuntos con imágenes en formato RGB), favorecen la aparición de estas colinas en las gráficas de error.

Finalmente, podemos concluir que, a medida que avanzan las épocas de entrenamiento, estas oscilaciones se vuelven cada vez menos notables, posiblemente porque el modelo encuentra cada vez soluciones más simples. Esto sugiere que, si se entrenara durante un mayor número de épocas de entrenamiento, estos picos podrían llegar a ser prácticamente imperceptibles o incluso desaparecer por completo.

## Glosario matemático

$\mathbb{R}$  Conjunto de los números reales.

$\mathbb{R}_+$  Conjunto de los números reales positivos.

$\mathbb{N}$  Conjunto de los números naturales.

**i.d.d.** Independientes e idénticamente distribuidas.

$\mathbb{E}[X]$  Esperanza de la variable aleatoria  $X$ .

$\mathbb{E}_{\mathcal{K}}[X]$  Esperanza de la variable aleatoria  $X$  con respecto al conjunto  $\mathcal{K}$ .

$P[A]$  Probabilidad de que ocurra el suceso  $A$ .

$P[X, Y]$  Probabilidad conjunta de las variables aleatorias  $X$  e  $Y$ .

$Var(X)$  Varianza de la variable aleatoria  $X$ .

$\mathcal{M}_{m \times n}(\mathbb{R})$  Espacio de matrices de tamaño  $m \times n$  con entradas en  $\mathbb{R}$ .

$rang(X)$  Rango de la matriz  $X$ .

$\lambda(A)$  Valor propio de la matriz  $A$ .

$\sigma(A)$  Valor singular de la matriz  $A$ .

**SVD** Descomposición en valores singulares.

$X^\dagger$  Pseudoinversa de la matriz  $X$ .

**OLS** Regresión lineal de mínimos cuadrados.

$H_{\mathcal{F}}$  Matriz hessiana de la función  $\mathcal{F}$ .

$|G|$  Cardinalidad del conjunto  $G$ .

$\#G$  Cardinalidad del conjunto  $G$  (notación alternativa para cuando es finita).

$\nabla \mathcal{L}$  Gradiente de la función  $\mathcal{L}$ .

$\mathcal{D}\mathcal{F}$  Derivada de la función  $\mathcal{F}$ .

$\mu - PL$  Condición modificada de Polyak y Lojasiewicz.

**Esencialmente no convexo** Funciones cuya forma no es convexa en ningún entorno de un minimizador global.

$K(w)$  Núcleo tangente en el vector  $w$ .

$N_{eff}(A)$  Dimensionalidad efectiva de la matriz  $A$ .

#### D. Irregularidades en la Dinámica del Error

$\langle \cdot, \cdot \rangle$  Producto interno.

$E_n(f)$  Error de aproximación lineal.

$\sigma_n(f)$  Error de aproximación no lineal.

$\mathcal{A}^\alpha$  Espacio de aproximación (funciones) cuyo error de aproximación  $\sigma_n(f)$  está acotado superiormente por un múltiplo de  $n^{-\alpha}$ .

**Biblioteca** En aproximación no lineal, conjunto de bases.

**Diccionario** En aproximación no lineal, subconjunto arbitrario de un espacio de Hilbert.

$\sigma_n^{\mathcal{L}}(f)$  Error de aproximación con respecto a la biblioteca  $\mathcal{L}$ .

$\sigma_n(\mathbb{D}, \mathbb{D})$  Error de aproximación con respecto al diccionario  $\mathbb{D}$ .

$\mathcal{K}_\tau^o(\mathbb{D}, M)$  Clase de funciones construidas a partir de combinaciones lineales de un conjunto finito, cuyos coeficientes están acotados por una constante que depende de  $\tau$  y de  $M$ .

$\mathcal{K}_\tau(\mathbb{D})$  Unión de las clases  $\mathcal{K}_\tau(\mathbb{D}, M)$  para  $M > 0$ .

## Glosario informático

**DDD** Deep Double Descent.

$\mathcal{X}$  Espacio muestral que contiene todas las entradas posibles para un modelo.

$\mathcal{Y}$  Conjunto compuesto por todas las posibles salidas del modelo.

$\mathcal{L}$  Función de pérdida a minimizar por el modelo.

$\mathcal{D}$  Conjunto de datos de entrenamiento.

$\mathcal{H}$  Conjunto de hipótesis formado por las hipótesis candidatas y la función objetivo.

$\mathcal{A}$  Algoritmo de aprendizaje.

$f$  Función objetivo.

$\bar{g}$  Hipótesis promedio o ideal.

**Sesgo** Error producido por la diferencia entre la hipótesis ideal y la función objetivo.

**Varianza** Error producido por la diferencia entre la hipótesis elegida y la hipótesis ideal.

**Descenso de gradiente** Método de optimización para minimizar una función de pérdida.

$\eta$  Tasa de aprendizaje o *learning rate*.

$E_{out}$  Error fuera de la muestra o error de generalización.

$E_{in}$  Error dentro de la muestra o error de entrenamiento.

**Sweet spot** Mínimo del error de generalización según la sabiduría clásica (tras el primer descenso).

$m_{\mathcal{H}}(n)$  Función de crecimiento para el conjunto  $\mathcal{H}$ .

$d_{vc}$  Dimensión de Vapnik-Chervonenkis.

$\mathcal{R}(\mathcal{K})$  Complejidad de Rademacher del conjunto  $\mathcal{K}$ .

**Complejidad efectiva del modelo (EMC)** Máximo número de ejemplos de entrenamiento en el que el modelo obtiene un  $E_{in}$  próximo a cero.

**Región moderna o de interpolación** Zona en la que la capacidad del modelo supera su complejidad efectiva.

**Umbral de interpolación** Punto en el que el modelo alcanza su complejidad efectiva.

**Principio de Ockham** Principio filosófico que favorece las explicaciones más simples entre las posibles.

#### *D. Irregularidades en la Dinámica del Error*

$w^*$  Solución óptima del problema de minimización de la función de pérdida.

**Sesgo inductivo** Conjunto de supuestos que utiliza un algoritmo de aprendizaje para preferir unas hipótesis sobre otras.

**Complejidad de Kolmogorov** Longitud del programa más corto que genera una hipótesis.

**Conexión residual** Técnica que permite que la entrada pueda saltar capas de una red.

## Bibliografía

- [AMMIL12] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, y Hsuan-Tien Lin. *Learning from Data: A Short Course*. AMLBook, 1st edición, 2012.
- [AS17] Madhu S. Advani y Andrew M. Saxe. High-dimensional dynamics of generalization error in neural networks, 2017. Preprint. arXiv: 1710.03667.
- [BB18] Randall Balestriero y Richard Baraniuk. A spline theory of deep learning. *Proceedings of the International Conference on Machine Learnin (PMLR)*, 80:374–383, 2018.
- [BB24] Christopher M. Bishop y Hugh Bishop. *Deep Learning: Foundations and Concepts*. Springer, 1st edición, 2024.
- [BD09] Shai Ben-David. *Theory-Practice Interplay in Machine Learning - Emerging Theoretical Challenges*. Springer, 1st edición, 2009.
- [BEHW87] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, y Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377–378, 1987.
- [Bel21] Mikhail Belkin. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation, 2021. Preprint. arXiv: 2105.14368.
- [BHMM19] Mikhail Belkin, Daniel Hsu, Siyuan Ma, y Soumik Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. *Proceedings of the National Academy of Sciences (PNAS)*, 116(32):15849–15854, 2019.
- [Bis95] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1st edición, 1995.
- [Biso06] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 1st edición, 2006.
- [Blu21] Avrim Blum. Mathematical toolkit. TTI-Chicago, Course Website, 2021. Lecture 5, Available at: <https://home.ttic.edu/~avrim/Toolkit21/15%20-%20real%20spectral%20theorem.pdf>.
- [Bry95] Włodzimierz Bryc. *The Normal Distribution: Characterizations with Applications*. Springer, 1st edición, 1995.
- [CJvdS23] Alicia Curth, Alan Jeffares, y Mihaela van der Schaar. A u-turn on double descent: Rethinking parameter counting in statistical learning, 2023. Preprint. arXiv: 2310.18988.
- [CMBK21] Lin Chen, Yifei Min, Mikhail Belkin, y Amin Karbasi. Multiple descent: Design your own generalization curve, 2021. Preprint. arXiv: 2008.01036.
- [CRF<sup>+</sup>25] Ben Cottier, Robi Rahman, Loredana Fattorini, Nestor Maslej, Tamay Besiroglu, y David Owen. The rising costs of training frontier ai models, 2025. Preprint. arXiv: 2405.21015.
- [Dem14] Amir Dembo. *Probability Theory: STAT310/MATH230*. CreateSpace Independent Publishing Platform, 1st edición, 2014.
- [DeV98] Ronald A. DeVore. Nonlinear approximation. *Acta Numerica*, 7:51–150, 1998.
- [DLK23] Xander Davies, Lauro Langosco, y David Krueger. Unifying grokking and double descent, 2023. Preprint. arXiv: 2303.06173.
- [DMPHO23] Radosvet Desislavov, Fernando Martínez-Plumed, y José Hernández-Orallo. Trends in ai inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustainable Computing: Informatics and Systems*, 38:100857, 2023.

## Bibliografía

- [Dom00] Pedro Domingos. A unifeid bias-variance decomposition and its applications. *Proceedings of the International Conference on Machine Learning (ICML)*, páginas 231–238, 2000.
- [dSB21] Stéphane d’Ascoli, Levent Sagun, y Giulio Biroli. Triple descent and the two kinds of overfitting: where and why do they appear? *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124002, 2021.
- [DT96] Ronald DeVore y V. Temlyakov. Some remarks on greedy algorithm. *Advances in Computational Mathematics*, 5:173–187, 1996.
- [Dui00] R.P.W. Duin. Classifiers in almost empty spaces. *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2:1–7, 2000.
- [EEAMT22] Mohamed Elasri, Omar Elharrouss, Somaya Ali Al-Maadeed, y Hamid Tairi. Image generation: A review. *Neural Processing Letters (NPL)*, 54:4609–4646, 2022.
- [FIS14] S.H. Friedberg, A.J. Insel, y L.E. Spence. *Linear Algebra*. Pearson Education, 1st edición, 2014.
- [GB10] Xavier Glorot y Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 9:249–256, 2010.
- [GBC16] Ian Goodfellow, Yoshua Bengio, y Aaron Courville. *Deep Learning*. MIT Press, 1st edición, 2016.
- [GBD92] Stuart Geman, Elie Bienenstock, y René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.
- [GFRW24] Micah Goldblum, Marc Finzi, Keefer Rowan, y Andrew Gordon Wilson. The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning, 2024. Preprint. arXiv: 2304.05366.
- [GK22] Philipp Grohs y Gitta Kutyniok. *Mathematical Aspects of Deep Learning*. Cambridge University Press, 1st edición, 2022.
- [GLSS19] Suriya Gunasekar, Jason Lee, Daniel Soudry, y Nathan Srebro. Implicit bias of gradient descent on linear convolutional networks, 2019. Preprint. arXiv: 1806.00468.
- [GSd<sup>+</sup>19] Mario Geiger, Stefano Spigler, Stéphane d’Ascoli, Levent Sagun, Marco Baity-Jesi, Giulio Biroli, y Matthieu Wyart. Jamming transition as a paradigm to understand the loss landscape of deep neural networks. *Physical Review E (PRE)*, 100(1), 2019.
- [HCB<sup>+</sup>19] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, y Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism, 2019. Preprint. arXiv: 1811.06965.
- [HEG<sup>+</sup>20] W. Ronny Huang, Zeyad Emam, Micah Goldblum, Liam Fowl, J. K. Terry, Furong Huang, y Tom Goldstein. Understanding generalization through visualizations, 2020. Preprint. arXiv: 1906.03291.
- [HT20] Fengxiang He y Dacheng Tao. Recent advances in deep learning theory, 2020. Preprint. arXiv: 2012.10931.
- [HTFo1] Trevor Hastie, Robert Tibshirani, y Jerome Friedman. *The Elements of Statistical Learning*. Springer, 1st edición, 2001.
- [HY20] Reinhard Heckel y Fatih Furkan Yilmaz. Early stopping in deep networks: Double descent and how to eliminate it, 2020. Preprint. arXiv: 2007.10099.
- [HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, y Jian Sun. Deep residual learning for image recognition, 2015. Preprint. arXiv: 1512.03385.

- [KB17] Diederik P. Kingma y Jimmy Ba. Adam: A method for stochastic optimization, 2017. Preprint. arXiv: 1412.6980.
- [KH91] Anders Krogh y John A. Hertz. A simple weight decay can improve generalization. *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, páginas 950–957, 1991.
- [KLW19] Uday Kamath, John Liu, y James Whitaker. *Deep Learning for NLP and Speech Recognition*. Springer, 1st edición, 2019.
- [KNH09] Alex Krizhevsky, Vinod Nair, y Geoffrey Hinton. Learning multiple layers of features from tiny images. *University of Toronto*, 2009.
- [Knio09] Oliver Knill. *Probability Theory and Stochastic Processes with Applications*. Overseas Press, 1st edición, 2009.
- [Kol56] A.N. Kolmogorov. *Foundations Of The Theory Of Probability*. Chelsea Publishing Company, 1st edición, 1956.
- [KSH12] Alex Krizhevsky, Ilya Sutskever, y Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, 1(9):1097–1105, 2012.
- [LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, y Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [LBD<sup>+</sup>89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, y L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.
- [LBH15] Yann LeCun, Yoshua Bengio, y Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.
- [LFK<sup>+</sup>24a] Sanae Lotfi, Marc Finzi, Sanyam Kapoor, Andres Potapczynski, Micah Goldblum, y Andrew Gordon Wilson. Pac-bayes compression bounds so tight that they can explain generalization, 2024. Preprint. arXiv: 2211.13609.
- [LFK<sup>+</sup>24b] Sanae Lotfi, Marc Finzi, Yilun Kuang, Tim G. J. Rudner, Micah Goldblum, y Andrew Gordon Wilson. Non-vacuous generalization bounds for large language models, 2024. Preprint. arXiv: 2312.17173.
- [LJY24] Fei-Fei Li, Justin Johnson, y Serena Yeung. Cs231n: Deep learning for computer vision. Stanford University, Course Website, 2024. Lecture 4, Slide 45, Available at: <http://cs231n.stanford.edu/>.
- [LLA22] Ivano Lauriola, Alberto Lavelli, y Fabio Aiolfi. An introduction to deep learning in natural language processing: Models, techniques, and tools. *Neurocomputing*, 470(C):443–456, 2022.
- [LSJR16] Jason D. Lee, Max Simchowitz, Michael I. Jordan, y Benjamin Recht. Gradient descent converges to minimizers, 2016. Preprint. arXiv: 1602.04915.
- [LT24] Marc Lafon y Alexandre Thomas. Understanding the Double Descent Phenomenon in Deep Learning, 2024. Preprint. arXiv: 2403.10459.
- [LXT<sup>+</sup>18] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, y Tom Goldstein. Visualizing the loss landscape of neural nets, 2018. Preprint. arXiv: 1712.09913.
- [LZB21] Chaoyue Liu, Libin Zhu, y Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks, 2021. Preprint. arXiv: 2003.00307.
- [Mal16] Stéphane Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203, 2016.

## Bibliografía

- [MBW20] Wesley J. Maddox, Gregory Benton, y Andrew Gordon Wilson. Rethinking parameter counting in deep models: Effective dimensionality revisited, 2020. Preprint. arXiv: 2003.02139.
- [McA99] David A. McAllester. Pac-bayesian model averaging. *Proceedings of the Annual Conference on Computational Learning Theory (COLT)*, páginas 164–170, 1999.
- [MCK20] Yifei Min, Lin Chen, y Amin Karbasi. The curious case of adversarially robust models: More data can help, double descent, or hurt generalization, 2020. Preprint. arXiv: 2002.11080.
- [MM18] Charles H. Martin y Michael W. Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning, 2018. Preprint. arXiv: 1810.01075.
- [MP67] V. A. Marchenko y L. A. Pastur. Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik*, 1(4):457–483, 1967.
- [MRVPL23] Chris Mingard, Henry Rees, Guillermo Valle-Pérez, y Ard A. Louis. Do deep neural networks have an inbuilt occam’s razor?, 2023. Preprint. arXiv: 2304.06670.
- [Mur22] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 1st edición, 2022.
- [Mur23] Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 1st edición, 2023.
- [NGLK18] Cláudia Neves, Ignacio Gonzalez, John Leander, y Raid Karoumi. A new approach to damage detection in bridges using machine learning. *Lecture Notes in Civil Engineering (LNCE)*, 5:73–84, 2018.
- [NKB<sup>+</sup>19] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, y Ilya Sutskever. Deep double descent: Where bigger models and more data hurt, 2019. Preprint. arXiv: 1912.02292.
- [NMB<sup>+</sup>19] Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, y Ioannis Mitliagkas. A modern take on the bias-variance tradeoff in neural networks, 2019. Preprint. arXiv: 1810.08591.
- [NRK21] Thao Nguyen, Maithra Raghu, y Simon Kornblith. Do wide and deep networks learn the same things? Uncovering how neural network representations vary with width and depth, 2021. Preprint. arXiv: 2010.15327.
- [Opp95] Manfred Opper. Statistical mechanics of learning: Generalization. *Springer*, páginas 922–925, 1995.
- [Opp01] Manfred Opper. Learning to generalize. *Academic Press*, 3, Part 2:763–775, 2001.
- [PBE<sup>+</sup>22] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, y Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022. Preprint. arXiv: 2201.02177.
- [PGMa19] Adam Paszke, Sam Gross, Francisco Massa, y et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, páginas 8024–8035, 2019.
- [Poo11] David Poole. *Álgebra lineal. Una introducción moderna*. Cengage Learning, 3rd edición, 2011.
- [Pre94] Roger S. Pressman. *Software Engineering: A Practitioner’s Approach*. McGraw-Hill, 1st edición, 1994.
- [Pri23] Simon J.D. Prince. *Understanding Deep Learning*. MIT Press, 1st edición, 2023.

- [RH21] Lars Ruthotto y Eldad Haber. An introduction to deep generative modeling, 2021. Preprint. arXiv: 2103.05180.
- [Rip96] Brian D. Ripley. *Pattern recognition and neural networks*. Cambridge University Press, 1st edición, 1996.
- [RSSW24] Hrithik Ravi, Clayton Scott, Daniel Soudry, y Yutong Wang. The implicit bias of gradient descent on separable multiclass data, 2024. Preprint. arXiv: 2411.01350.
- [Sah18] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way. <https://medium.com/towards-data-science/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>, 2018. [Recurso online, accedido el 19 de febrero de 2025].
- [SBD<sup>+</sup>19] Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, y David Daniel Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment (JSTAT)*, 2019, 2019.
- [Sch15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61(2065):85–117, 2015.
- [Sej20] Terrence J. Sejnowski. The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences*, 117:30033–30038, 2020.
- [SFB<sup>+</sup>22] Gowthami Somepalli, Liam Fowl, Arpit Bansal, Ping Yeh-Chiang, Yehuda Dar, Richard Baraniuk, Micah Goldblum, y Tom Goldstein. Can neural nets learn the same model twice? investigating reproducibility and double descent from the decision boundary perspective, 2022. Preprint. arXiv: 2203.08124.
- [SGd<sup>+</sup>19] S Spigler, M Geiger, S d’Ascoli, L Sagun, G Biroli, y M Wyart. A jamming transition from under- to over-parametrization affects generalization in deep learning. *Journal of Physics A: Mathematical and Theoretical*, 52(47), 2019.
- [SGM20] Emma Strubell, Ananya Ganesh, y Andrew McCallum. Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(9):13693–13696, 2020.
- [SHN<sup>+</sup>24] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, y Nathan Srebro. The implicit bias of gradient descent on separable data, 2024. Preprint. arXiv: 1710.10345.
- [SKR<sup>+</sup>23] Rylan Schaeffer, Mikail Khona, Zachary Robertson, Akhilan Boopathy, Kateryna Pistunova, Jason W. Rocks, Ila Rani Fiete, y Oluwasanmi Koyejo. Double descent demystified: Identifying, interpreting & ablating the sources of a deep learning puzzle, 2023. Preprint. arXiv: 2303.14151.
- [SLHS22] Sidak Pal Singh, Aurelien Lucchi, Thomas Hofmann, y Bernhard Schölkopf. Phenomenology of double descent in finite-width neural networks, 2022. Preprint. arXiv: 2203.07337.
- [SLJ<sup>+</sup>14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, y Andrew Rabinovich. Going deeper with convolutions, 2014. Preprint. arXiv: 1409.4842.
- [Sol64] R.J. Solomonoff. A formal theory of inductive inference. part i. *Information and Control*, 7(1), 1964.
- [Str23] Gilbert Strang. *Introduction to Linear Algebra*. CUP, 6th edición, 2023.
- [Swaz20] Swapna. Convolutional Neural Network | Deep Learning. <https://developersbreach.com/convolution-neural-network-deep-learning/>, 2020. [Recurso online, accedido el 19 de febrero de 2025].
- [TGLM22] Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, y Gabriel F. Manso. The computational limits of deep learning, 2022. Preprint. arXiv: 2007.05558.

## Bibliografía

- [Val84] L. G. Valiant. A theory of the learnable. *Association for Computing Machinery*, 27(11):1134–1142, 1984.
- [Vap91] V. Vapnik. Principles of risk minimization for learning theory. *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, 4:831–838, 1991.
- [VCR89] F. Vallet, J.-G. Cailton, y Ph Refregier. Linear and nonlinear extension of the pseudo-inverse solution for learning boolean functions. *Europhysics Letters*, 9(4):315, 1989.
- [WH88] Bernard Widrow y Marcian E. Hoff. Adaptive switching circuits. *Neurocomputing*, 1:96–104, 1988.
- [Wil25] Andrew Gordon Wilson. Deep learning is not so mysterious or different, 2025. Preprint. arXiv: 2503.02113.
- [YS24] Tian-Le Yang y Joe Suzuki. Dropout drops double descent, 2024. Preprint. arXiv: 2305.16179.
- [YYY<sup>+</sup>20] Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt, y Yi Ma. Rethinking bias-variance trade-off for generalization of neural networks, 2020. Preprint. arXiv: 2002.11328.
- [ZBH<sup>+</sup>21] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, y Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.