



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

Tutores:
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.

Responsable de tutorización	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 27 de marzo de 2025

Fdo: Juan Antonio Ruiz Arévalo

Dicatoria

Agradecimientos

Agradecimientos

Resumen

Summary

Índice general

Agradecimientos	III
Resumen	V
Summary	VI
Índice de figuras	XI
Índice de tablas	XII
Introducción	XIV
1. Definición del problema	XV
2. Motivación	XVI
3. Objetivos	XVII
3.1. Objetivo matemático	XVIII
3.2. Objetivo informático	XVIII
4. Planificación del proyecto	XVIII
I. Fundamentos Teóricos	1
1. Probabilidad	3
1.1. Espacios de probabilidad y σ -álgebras	3
1.2. Variables aleatorias y esperanza	4
1.2.1. Probabilidad condicional	6
1.2.2. Independencia de variables aleatorias	8
1.2.3. Propiedades de la esperanza y varianza	12
1.3. Distribuciones de probabilidad	14
1.3.1. Distribución Normal	14
2. Descomposición en valores singulares y pseudoinversa de una matriz	16
2.1. Vectores y matrices	16
2.2. SVD y pseudoinversa	18
3. Aprendizaje Automático y Aprendizaje Profundo	23
3.1. Fundamentos	23
3.2. Redes neuronales artificiales	24
3.2.1. Redes neuronales convolucionales	25
4. El dilema clásico del aprendizaje	31
4.1. Concepto de aprendizaje	31
4.1.1. Descenso de gradiente y aprendizaje	32

4.2.	Bias-variance tradeoff	35
4.2.1.	Formulación matemática del E_{out}	36
4.3.	Equilibrio clásico entre sesgo y varianza	39
4.3.1.	Curva de aprendizaje	40
4.4.	Underfitting y overfitting	41
II.	Estado del Arte	46
5.	Trabajos relacionados	48
5.1.	Origen y primeras manifestaciones	49
5.2.	El nacimiento del Deep Double Descent	50
5.3.	Avances recientes	51
III.	Desarrollo Teórico y Empírico	54
6.	Análisis teórico del Deep Double Descent	56
6.1.	Planteamiento teórico	56
6.1.1.	Análisis intuitivo en un problema de mínimos cuadrados	58
6.2.	Sesgo inductivo del descenso de gradiente	63
6.2.1.	Problema de regresión	63
6.2.2.	Problema de clasificación con datos separables	67
6.3.	Optimización en la zona sobreparametrizada	69
6.3.1.	Elección de la mejor hipótesis	74
6.4.	Resto de desarrollos a realizar	76
6.5.	Aproximación no lineal	76
6.5.1.	Analogía con el deep double descent	76
6.6.	Conclusión	76
7.	Análisis empírico del Deep Double Descent	77
7.1.	Materiales y métodos	77
7.1.1.	Datasets	77
7.1.2.	Arquitecturas utilizadas	79
7.1.3.	Hiperparámetros	82
7.2.	Experimentos	82
7.2.1.	Aproximación polinómica de Legendre	82
7.2.2.	Noise-wise double descent	84
7.2.3.	Sample-wise double descent	85
7.3.	Conclusión	86
IV.	Conclusiones y Trabajos Futuros	87
8.	Conclusiones	89
9.	Trabajos futuros	90
A.	Apéndice	91

Índice general

B. Apéndice	93
C. Apéndice	95
Glosario	97
Bibliografía	99

Índice de figuras

1.	Ejemplo de doble descenso profundo en ResNet18 [NKB ⁺ 19].	xv
1.1.	Ejemplos de distribuciones normales.	15
3.1.	Ejemplos de problemas de clasificación y regresión.	23
3.2.	Ejemplos de neurona biológica [NGLK18] y neurona artificial (basada en [LJY24]).	24
3.3.	Ejemplo de CNN utilizada para clasificación de imágenes [Swa20].	26
3.4.	Ejemplo de convolución con padding [Sah18].	27
3.5.	Ejemplos de pooling utilizados en CNN.	28
3.6.	Ejemplos de funciones de activación utilizadas en CNN.	30
4.1.	Diagrama representando el concepto básico de aprendizaje.	32
4.2.	Distintas tasas de aprendizaje para el descenso de gradiente [AMMIL12].	33
4.3.	Proceso de retropropagación del error [Biso06].	34
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 bias/variance con underfitting y overfitting en el contexto clásico.	44
5.1.	Número de publicaciones relativas al Deep Double Descent (ir actualizando histograma de cara a nuevos papers).	48
5.2.	Deep Double Descent presente en ADALINE.	49
5.3.	Curva del error unificada entre la teoría clásica y moderna [BHMM19].	51
6.1.	Ejemplo de doble descenso con las distintas zonas de parametrización.	57
6.2.	Ejemplos de distribuciones de Marchenko-Pastur [MM18].	62
6.3.	Ejemplos de paisajes de la función de pérdida en redes neuronales [LZB21].	70
6.4.	Ejemplo de función de pérdida que satisface la condición μ -PL [LZB21].	74
6.5.	Distintos modelos polinómicos para aproximar una función.	75
7.1.	Arquitectura ResNet18 modificada.	80
7.2.	Arquitectura 3CNN.	81
7.3.	Intuición del <i>Deep Double Descent</i> usando regresión polinómica.	83
7.4.	Doble descenso al utilizar aproximación polinómica de Legendre.	83
7.5.	Doble descenso para distintos niveles de ruido.	84
7.6.	Umbral de interpolación para el doble descenso con distintos niveles de ruido.	85
7.7.	Ejemplo de <i>sample-wise double descent</i>	86
C.1.	Doble descenso para distintos tamaños de lote.	95
C.2.	Doble descenso para distintas tasas de aprendizaje.	96

Índice de tablas

1.	Planificación temporal inicial del proyecto.	xx
2.	Estimación del coste del proyecto.	xx
5.1.	Resumen de los principales artículos junto con sus contribuciones.	53
6.1.	76
7.1.	Resumen de los datasets utilizados.	78
7.2.	Resumen de las arquitecturas 2NN.	79
7.3.	Número de parámetros de las arquitecturas ResNet18 modificadas.	81
7.4.	Número de parámetros de las arquitecturas 3CNN.	81
7.5.	Resumen de los experimentos para el doble descenso por nivel de ruido.	85
7.6.	Resumen de los experimentos para el doble descenso por número de ejemplos de entrenamiento.	86
C.1.	Resumen de los experimentos realizados para el doble descenso con distinto tamaño de lote.	95
C.2.	Resumen de los experimentos realizados para el doble descenso con distintas tasas de aprendizaje.	96

Introducción

En los últimos años, el *Deep Double Descent* ha surgido como un importante campo de interés en el aprendizaje automático. La sabiduría tradicional sugiere que, a medida que aumenta la complejidad del modelo, el error fuera de la muestra disminuye inicialmente hasta alcanzar un mínimo y, a continuación, aumenta debido al sobreajuste, formando la tradicional curva con forma de “U”. Este concepto tradicional del aprendizaje automático, conocido como equilibrio entre sesgo y varianza, difiere de las recientes observaciones obtenidas, que cuestionan este punto de vista, especialmente en modelos de aprendizaje profundo, en los que puede producirse una segunda disminución del error fuera de la muestra y alcanzar un nuevo mínimo, formando una nueva curva en la gráfica del error de generalización que presenta dos descensos. Este novedoso descubrimiento pone en tela de juicio la sabiduría clásica sobre el tema y proporciona nuevas perspectivas a la hora de crear y entrenar los modelos.

Este trabajo de fin de grado se centra en explorar el concepto del *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 párametros 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 nuevos conocimientos 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 es significativamente complejo, 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 el suceso, 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. Con este trabajo buscamos cerrar la brecha entre la teoría y la práctica unificando explicaciones teóricas suficientemente rigurosas con ejemplos empíricos del mundo real con el objetivo de ofrecer una comprensión lo más completa posible.

1. Definición del problema

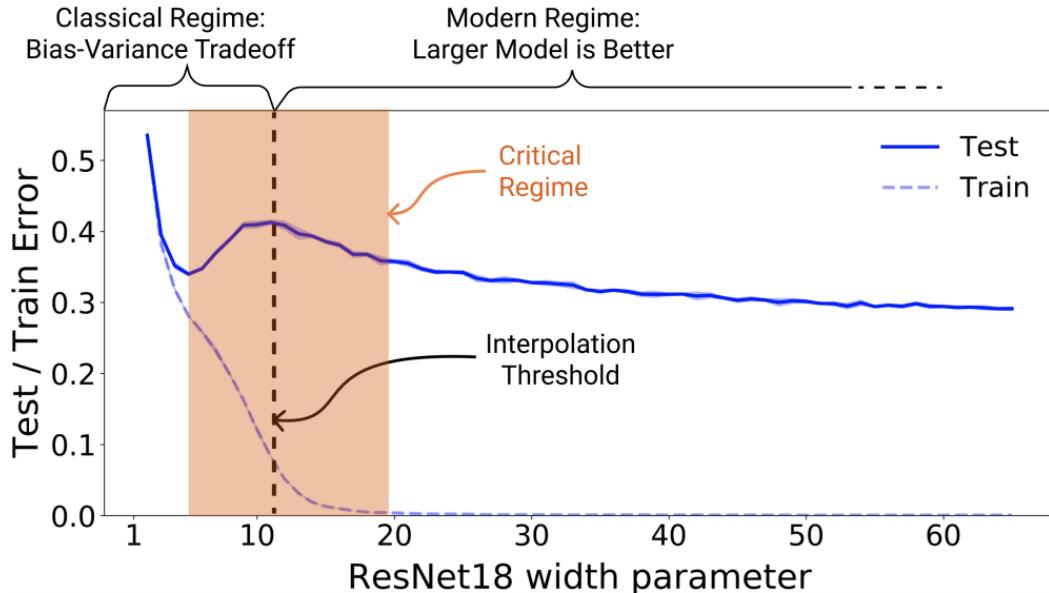


Figura 1.: Ejemplo de doble descenso en ResNet18 [NKB⁺19]. La imagen muestra el error de entrenamiento (*train error*) y de generalización (*test error*) para la arquitectura ResNet18 con diferente capacidad (número de parámetros). En ella, observamos las tres regiones de actuación del modelo claramente diferenciadas, así como el máximo del error de generalización correspondiente al umbral de interpolación y los dos descensos de la curva 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) de un modelo de aprendizaje como función de la capacidad del mismo. De manera intuitiva, podemos distinguir 3 zonas o regiones diferenciadas en dicha curva:

- **Región clásica (infraparametrizada):** En esta región, el modelo no es capaz de 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) dando lugar a la clásica curva en forma de “U”, relacionado con el hecho de que el modelo, en lugar de seguir aprendiendo de los datos y obtener patrones de los mismos, memoriza los datos de entrenamiento.
- **Región moderna (sobreparametrizada):** En esta región, 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 régión 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 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 un nuevo mínimo en la curva del error de generalización, es decir, 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 que tendremos que replantearnos algunas respuestas, tradicionalmente correctas, ante preguntas clásicas del aprendizaje profundo.

2. Motivación

En el ámbito del aprendizaje automático, es de cultura general conocer la existencia de 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, BGKP22].

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⁺21, Mal16, Pri23, BB23, BGKP22, Brb18, SBD⁺18], 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 del *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. Sin embargo, con la aparición del *Deep Double Descent*, se ha demostrado que, más allá del sobreajuste, los modelos no solo

continúan mejorando su capacidad de generalización, logrando mejores predicciones, sino que incluso mejoran, alcanzando una generalización aún mejor a la inicial.

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 deep learning [TGLM22, CRF⁺25]. Por ello, en la actualidad, aún se aplican técnicas de regularización 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 [Opp95, Opp01], 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. Además, sentaron las bases para la comprensión moderna del suceso observado en modelos profundamente sobreparametrizados.

No obstante, no es hasta 2019 cuando Belkin et. al en [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 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 y fundamentan la implementación práctica, mientras que los resultados experimentales permiten validar y enriquecer la comprensión teórica. A su vez, cada una de estas líneas 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 de 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 su 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 y representaciones experimentales 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 4 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.

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.
- 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 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.

De este modo, se puede observar la planificación inicial del proyecto en la Tabla 1.

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 es de 35€/hora. A esta

Introducción

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	09	
Analisis de requisitos	5 - 150																						
Diseño	4 - 120																						
Implementación	3 - 90																						
Pruebas	8 - 240																						

Tabla 1.: Planificación temporal inicial del proyecto.

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. El desglose detallado de estos costes se puede consultar en la Tabla 2.

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€.

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

Item	Costo
Salario	16 800.00€
Portátil de Gama Media	1 000.00€
Servidor GPU	2 186.52€
Otros	300.00€
Total	20 286.52€

Tabla 2.: Estimación del coste del proyecto.

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] y [Knio9].

1.1. Espacios de probabilidad y σ -álgebras

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

Definición 1.1 (σ -álgebra). Un conjunto \mathcal{A} de subconjuntos de Ω ($\mathcal{A} \subseteq \mathcal{P}(\Omega)$) se dirá que es una σ -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 σ -álgebras es una σ -álgebra.

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

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

Definición 1.3 (Medida de probabilidad). Dado un espacio de medida (Ω, \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. $P[\Omega] = 1$,

1. Probabilidad

3. P es σ -aditiva, es decir, si $A_n \in \mathcal{A}$ con $n \in \mathbb{N}$ son conjuntos disjuntos dos a dos, entonces

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

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 (Ω) 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 (Ω, \mathcal{A}) , llamaremos *espacio de probabilidad* a la tripleta (Ω, \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 (Ω_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 σ -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 el valor de la cara superior del dado cuando se lanza. 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 una función $F_X : \mathbb{R} \rightarrow [0, 1]$ dada 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:

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

1. Probabilidad

Las probabilidades asociadas con todos los posibles resultados del experimento deben ser no negativas y sumar 1, 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, 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 (Ω, \mathcal{A}, P) . Definimos la *esperanza o valor esperado* de X , denotado 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:

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

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 viene definida por:

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

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. 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 Ω 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:

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

Usando 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:

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

Finalmente, usando la Definición 1.20, obtenemos que

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

1. Probabilidad

Por tanto, 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 las 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).

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

1. Probabilidad

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

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 1, 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 Lebesgue 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

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

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

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

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 , $i \in \{1, \dots, n\}$ 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 , $i \in \{1, \dots, n\}$ de la siguiente manera:

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

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 , $i \in \{1, \dots, n\}$ de la siguiente manera:

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

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

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

Definición 1.32. Sean X_1, \dots, X_n variables aleatorias definidas sobre el mismo espacio de probabilidad (Ω, \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

1. Probabilidad

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 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.

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

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

□

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.

□

Teorema 1.37. Sean X_1, \dots, X_n variables aleatorias independientes definidas sobre el mismo espacio de probabilidad (Ω, \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 desenlace 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 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 (Ω, \mathcal{A}, P) hasta el espacio de medida $(\mathbb{R}, \mathcal{B})$.

□

Observación 1.39. Sea 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

1. Probabilidad

la función de densidad.

Es conveniente remarcar la observación anterior, pues 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 ($\text{Var}(X)$) al valor esperado dado por

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

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

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

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

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. En esta sección trataremos algunas de las distribuciones más usuales al trabajar con datos del mundo real y que, además, serán utilizadas durante el transcurso del trabajo.

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.

Definición 1.42 (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 $X \sim \mathcal{N}(\mu, \sigma)$. 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 1.43. Si $X \sim \mathcal{N}(\mu, \sigma)$, entonces la distribución es simétrica con respecto a μ .

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

Proposición 1.44. Sea $X \sim \mathcal{N}(\mu, \sigma)$. Entonces

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

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

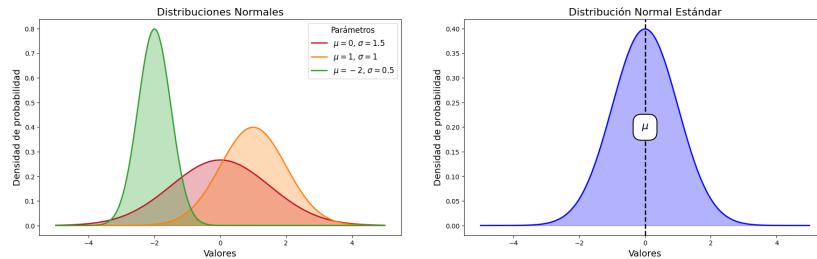


Figura 1.1.: Ejemplos de distribuciones normales. A la izquierda observamos distintas distribuciones normales, donde podemos notar como el parámetro μ determina el centro de la distribución y el parámetro σ controla la dispersión de los datos. A la derecha observamos una distribución normal estándar, donde se puede apreciar con claridad que la distribución es simétrica con respecto a μ . Imagen original del autor.

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] y [Poo11]. Estos libros nos ayudaron 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 δ_{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 $(I_n)_{ij} = \delta_{ij}$.

Definición 2.2. Sea A una matriz de tamaño $m \times n$. Se dice que A es una matriz **escalonada** si es O o 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

2. Descomposición en valores singulares y pseudoinversa de una matriz

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 λ se llama **valor propio** (o autovalor) de la matriz A correspondiente al vector propio v . Recordemos que λ 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 Σ 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 .

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

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

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 . Entonces se tiene que U es una matriz 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

2. Descomposición en valores singulares y pseudoinversa de una matriz

tanto,

$$\begin{aligned}
 AV &= A [\mathbf{v}_1 \ \cdots \ \mathbf{v}_n] \\
 &= [A\mathbf{v}_1 \ \cdots \ A\mathbf{v}_n] \\
 &= [A\mathbf{v}_1 \ \cdots \ A\mathbf{v}_r \ 0 \ \cdots \ 0] \\
 &= [\sigma_1 \mathbf{u}_1 \ \cdots \ \sigma_r \mathbf{u}_r \ 0 \ \cdots \ 0] \\
 &\quad \left[\begin{array}{cccc} \sigma_1 & \cdots & 0 & O \\ \vdots & \ddots & \vdots & O \\ 0 & \cdots & \sigma_r & O \\ O & O & O & O \end{array} \right] \\
 &= [\mathbf{u}_1 \ \cdots \ \mathbf{u}_m] U\Sigma
 \end{aligned}$$

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 donde las columnas de A 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 $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$. Sea Σ^\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 Σ^\dagger es la pseudoinversa de Σ . 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 $\text{Im}(A)$ y $\text{Im}(A)^T$ respectivamente. Además, si A es de rango completo, es decir, $\text{rango}(A) = r = \min\{m, n\}$, 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. $\text{Im}(I - A^\dagger A) = \ker(A)$.
2. $\ker(A^\dagger) = \ker(A^T)$.

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

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

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

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 $A^\dagger = V\Sigma^+ U^T$. Sean $y = V^T x$ y $c = U^T b$, expresados de la siguiente forma

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

donde $y_1, c_1 \in \mathbb{R}^r$.

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

2. Descomposición en valores singulares y pseudoinversa de una matriz

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

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 \mathbf{x} de mínimos cuadrados son de la forma

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

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

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

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

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

□

3. Aprendizaje Automático y Aprendizaje Profundo

3.1. Fundamentos

El aprendizaje automático o *machine learning* (ML) ([Biso6], [Mur22] y [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*.

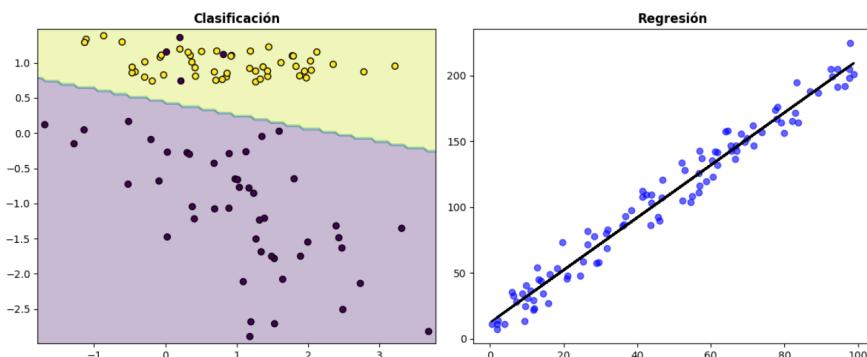


Figura 3.1.: Ejemplos de problemas de clasificación y regresión. A la izquierda, se muestra un problema de clasificación binaria, donde se busca encontrar el hiperplano (línea verde) que separe ambos conjuntos de datos etiquetados. A la derecha, se muestra un problema de regresión, donde se busca encontrar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

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

3. Aprendizaje Automático y Aprendizaje Profundo

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 centraremos en la clasificación de imágenes.

El aprendizaje profundo o *deep learning* (DL) ([BB23], [Pri23] y [LBH15]) representa un área del aprendizaje automático que utiliza redes neuronales artificiales, inspiradas en la estructura y función del cerebro humano, con múltiples capas, conocidas como redes neuronales profundas ([GBC16] y [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.

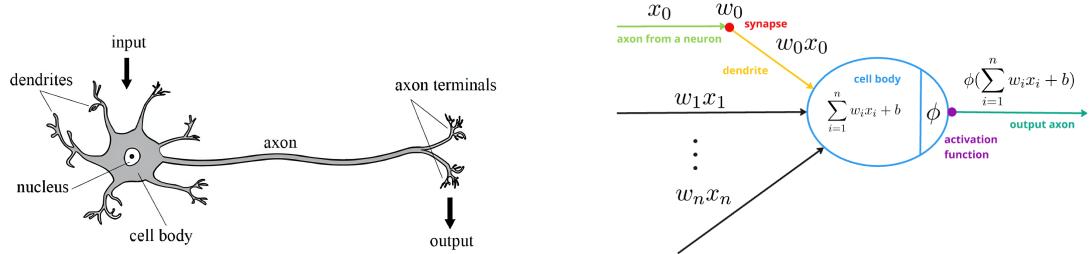


Figura 3.2.: Ejemplos de una neurona biológica [NGLK18] (izquierda) y una neurona artificial (derecha), basada en [LJY24].

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 que se trate 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 un problema de 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 tanto, 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 sea lo suficientemente representativo, ya que de este conjunto se extraerán los patrones relevantes que la red necesitará aprender.

3.2.1. Redes neuronales convolucionales

Las redes neuronales convolucionales o *convolutional neural networks* (CNN) ([LBD⁺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 redes neuronales convolucionales 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 de aprendizaje jerárquico de características 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 redes neuronales convolucionales 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*, respon-

3. Aprendizaje Automático y Aprendizaje Profundo

sables de la reducción de la dimensionalidad de la entrada sin perder las características importantes y las capas totalmente conectadas o *fully connected*, que se encuentran en la parte final de la red y que permiten combinar de manera efectiva las características extraídas por las capas anteriores para realizar la predicción final (véase Figura 3.3). Estas capas operan de manera conjunta, transformando la entrada y extrayendo progresivamente las características más relevantes, las cuales se van refinando y volviéndose más abstractas conforme se avanza por la red.

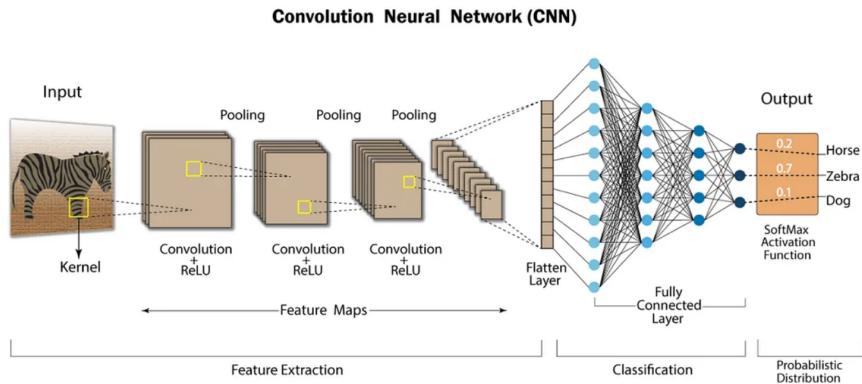


Figura 3.3.: Ejemplo de CNN utilizada para clasificación de imágenes [Swa20]. La entrada a la red es una imagen en formato RGB. La extracción de características se realiza mediante varias capas convolucionales, seguidas de funciones de activación y capas de pooling, las cuales ayudan a reducir la dimensionalidad. Posteriormente, las características extraídas se aplanan (*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.

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.

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 resultante. Así, a mayor cantidad de filtros, la profundidad del mapa de características resultantes también es mayor.

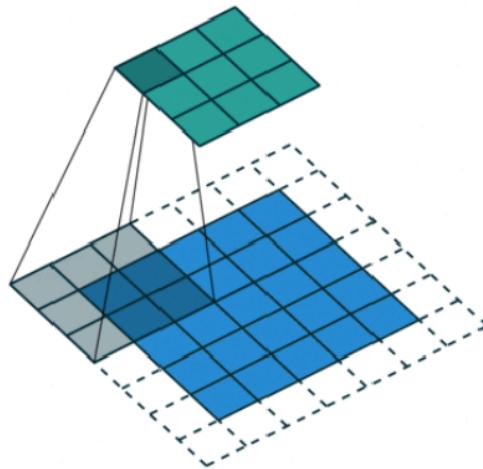


Figura 3.4.: Ejemplo de convolución con padding [Sah18]. El color azul hace referencia a la entrada, mientras que el color verde denota el resultado de la convolución. En particular, el color verde oscuro destaca el resultado de la convolución que se realiza sobre la zona grisácea de la imagen, utilizando,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

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

3. Aprendizaje Automático y Aprendizaje Profundo

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, lo que 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 max pooling, integrado en algunas de las arquitecturas que utilizamos para los experimentos.

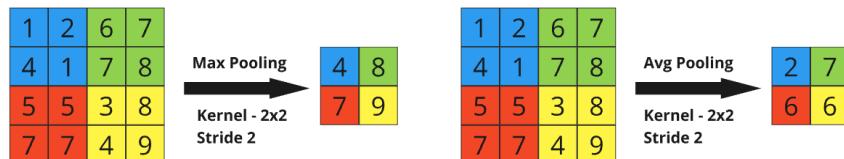


Figura 3.5.: Ejemplos de pooling utilizados en CNN. A la izquierda, se muestra el max pooling, donde se selecciona el valor máximo de una determinada región (en este caso 2x2). A la derecha, se muestra el average pooling, donde se selecciona el valor promedio de la región influida por el filtro. Imagen original del autor.

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. Incluso si solo se usaran funciones de activación lineal, la red neuronal se comportaría como una función lineal. Como menciona Bishop en su libro: “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” ([\[Biso6\]](#)), 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](#)).

3. Aprendizaje Automático y Aprendizaje Profundo

- **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. Por tanto, para el caso de $K = 2$ (clasificación binaria), esta función se reduce a la función sigmoide.

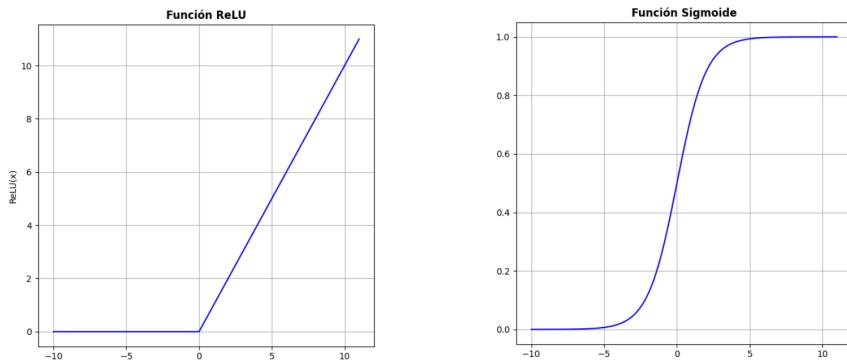


Figura 3.6.: Ejemplos de funciones de activación utilizadas en CNN. A la izquierda, se muestra la función ReLU, donde se observa como deja invariantes los valores positivos, estableciendo a cero los valores negativos. A la derecha, se muestra la función sigmoide, que mapea los valores de entrada a un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

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 harán entender de manera más precisa el suceso. 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 que aprender nada al conocer directamente la función objetivo, 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 x 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 objetivo $f : \mathcal{X} \rightarrow \mathcal{Y}$, que representa la función objetivo y 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 $x \in \mathcal{X}$ y $y \in \mathcal{Y}$, 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

4. El dilema clásico del aprendizaje

propósito de que la función candidata g siga replicando a la función objetivo f ante nuevos datos no disponibles.

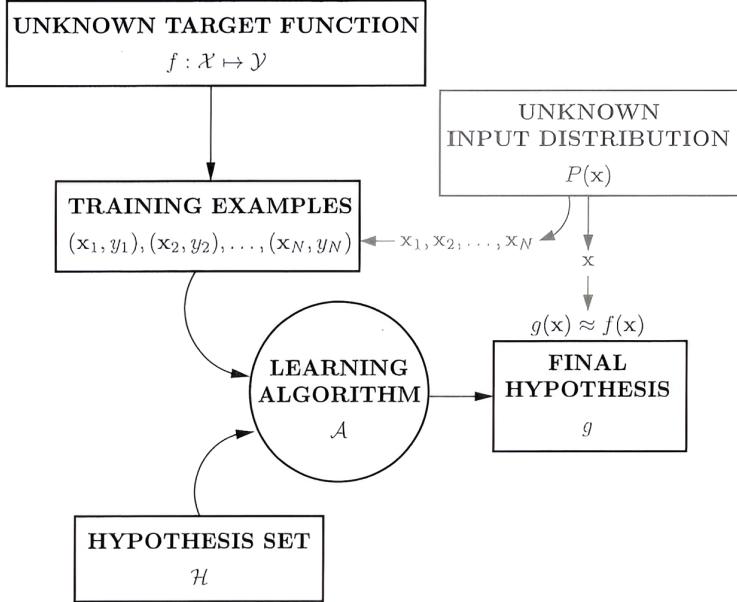


Figura 4.1.: Diagrama representando el concepto básico de aprendizaje [AMMIL12]. El algoritmo de aprendizaje (*learning algorithm*), utilizando los ejemplos de entrenamiento (*training examples*) muestreados de una distribución de probabilidad desconocida (*unknown input distribution*), buscará en el 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 la precisión y capacidad de generalización del modelo.

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 valores de los parámetros antes de la actualización, η 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 (η) 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), 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 el mínimo.

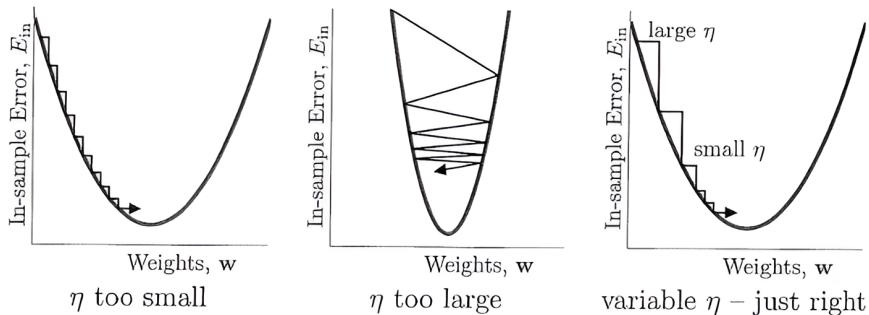


Figura 4.2.: Distintas tasas de aprendizaje para el descenso de gradiente [AMMIL12]. En la primera imagen, una tasa de aprendizaje pequeña lleva a una convergencia lenta con muchas actualizaciones. En la imagen central, una tasa demasiado grande provoca saltos bruscos que pueden impedir la convergencia. 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 sub-

4. El dilema clásico del aprendizaje

conjuntos 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. 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, 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 de coordenadas y , finalmente, se aplican las funciones de activación para introducir no linearidad. Este flujo de datos permite que el modelo genere una predicción para cada entrada.

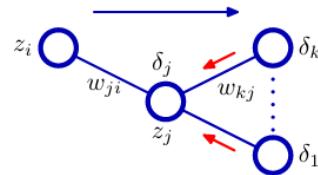


Figura 4.3.: Proceso de retropropagación del error [Biso6]. Dado un lote de entrenamiento, se propaga la información hacia delante (flecha azul), se calculan las activations y el error de salida. Seguidamente, se realiza el paso hacia atrás (flechas rojas), calculando el error δ_j de cada unidad j en 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, ϕ 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 \mathcal{D} como un subconjunto de \mathbb{R}^{D_i}) y w_{di} con $i \in \{0, 1, \dots, D_i\}$ representan 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 prueba o generalización (E_{out}) sobre datos desconocidos, lo que implicará que hemos conseguido aproximar de buena manera 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 f , al disponer de un mayor número de funciones candidatas. Sin embargo, 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 $g \in \mathcal{H}$.

Adicionalmente, el error de generalización E_{out} 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 del error. Sin embargo, cabe destacar que este ruido suele ser una limitación fundamental de la generalización del modelo.

4. El dilema clásico del aprendizaje

4.2.1. Formulación matemática del E_{out}

A continuación, detallaremos matemáticamente las componentes del error fuera de la muestra o de generalización. 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:

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

donde \mathbb{E}_x denota el valor esperado 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 tomado 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 del valor esperado 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. Sin embargo, 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 el valor esperado con respecto a \mathcal{D} , de la ecuación anterior tenemos que $(\bar{g}(x) - f(x))$ es una constante. Por tanto, dado que el valor esperado de una constante es la propia constante, para obtener el valor esperado del término cruzado solo necesitamos conocer el valor esperado 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:

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

4. El dilema clásico del aprendizaje

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

Observación 4.1. Cuando hay ruido presente en los datos de entrenamiento, es decir, $y(x) = f(x) + \epsilon(x)$ y, además, ϵ es una variable aleatoria con media 0 y varianza σ^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 ϵ 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 ϵ 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 ϵ , el tercer sumando no depende de \mathcal{D} y el segundo sumando puede expresarse de la siguiente forma:

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

donde el valor esperado con respecto a ϵ 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 expresarse de la siguiente forma

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

Finalmente, aplicando el valor esperado sobre x a la expresión anterior, obtenemos el resultado buscado. □

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 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 utilizando el error cuadrático.

- 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 un modelo.

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 \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, las hipótesis son menos sensibles a las variaciones en el conjunto de datos. Como resultado, al cambiar el conjunto de datos para seleccionar una hipótesis, 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.

De este modo, el equilibrio buscado en esta sección con objeto de minimizar el error fuera de la muestra viene ligado a la elección de un conjunto de hipótesis (\mathcal{H}) suficientemente complejo como para acercarnos a la función objetivo y suficientemente simple como para no disponer de una cantidad excesiva de hipótesis que induzcan un alto grado de varianza. Este equilibrio se podrá conseguir mediante diversas técnicas, como la regularización, que se explicarán en secciones posteriores.

4. El dilema clásico del aprendizaje

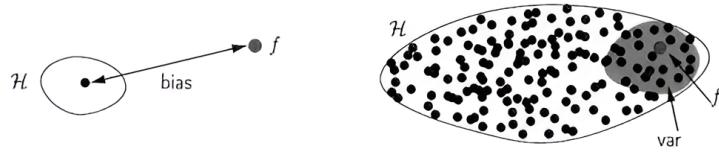


Figura 4.4.: Distintos casos del conjunto de hipótesis y de la función objetivo [AMMIL12]. A la izquierda observamos como nuestro conjunto de hipótesis presenta únicamente una función candidata, alejada de la función objetivo f , lo que implica un alto sesgo y una varianza nula. A la derecha observamos un conjunto de hipótesis con numerosas funciones candidatas que incluye a la función objetivo f , por lo que el sesgo es muy cercano a 0 y la varianza es grande.

4.3.1. Curva de aprendizaje

Para finalizar este capítulo, 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 $\mathbb{E}_{\mathcal{D}}[E_{out}(g^{(\mathcal{D})})]$, donde ambos errores son funciones del tamaño del conjunto de datos (N).

Denominaremos *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 (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**

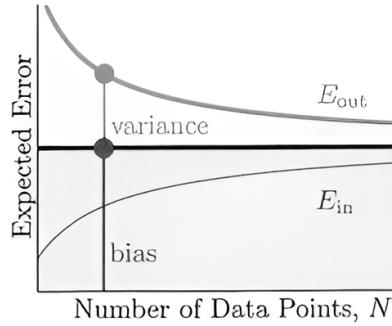


Figura 4.5.: Ejemplo de curva de aprendizaje tradicional [AMMIL12]. Observamos la curva de aprendizaje de un modelo con respecto al tamaño del conjunto de datos. Se puede comprobar como el E_{out} decrece, mientras que el E_{in} crece. Además, se puede apreciar la descomposición sesgo-varianza, donde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

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 o el número de épocas de entrenamiento del modelo y en el eje Y el error esperado (véase Figura 4.6), tanto dentro como fuera de la muestra, para un número fijo de datos de entrenamiento.

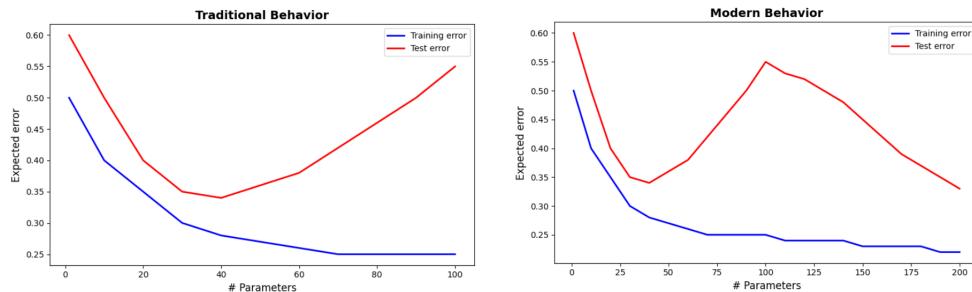


Figura 4.6.: Ejemplos de curvas de aprendizaje modificadas para este proyecto. A la izquierda se muestra la curva clásica de aprendizaje, donde el error de test aumenta llegado a un cierto punto. A la derecha, la curva moderna refleja que, al aumentar la complejidad del modelo, el error de test vuelve a disminuir. Imagen original del autor.

4.4. Underfitting y overfitting

Consideraremos 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

4. El dilema clásico del aprendizaje

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 $L(g(X), Y)$, 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, es decir, cuando el número de parámetros del modelo es significativamente menor que el número de datos de entrenamiento disponibles.

Definición 4.2 (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

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

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, es decir

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

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

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

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

$$\hat{g} = \arg \min_{g \in \mathcal{H}} L_{emp}(g). \tag{4.7}$$

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

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

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 esta manera, el algoritmo de aprendizaje \mathcal{A} definido por el principio de minimización del riesgo empírico consiste en resolver el problema de optimización dado por la ecuación (4.7).

Proposición 4.4. *Para cualquier minimizador del riesgo empírico \hat{g} , el error de estimación verifica*

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

Demostración. Partimos de la siguiente desigualdad

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

con el primer sumando verificando

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

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

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

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

De este modo, la estrategia 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 4.4 (máxima brecha de generalización sobre \mathcal{H}) aumentará y la función candidata \hat{g} elegida como minimizado-

4. El dilema clásico del aprendizaje

ra 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. Además, es posible establecer de manera clara la 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 prueba, ligado a un alto sesgo. 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, ligado a una varianza elevada y un sesgo bajo.

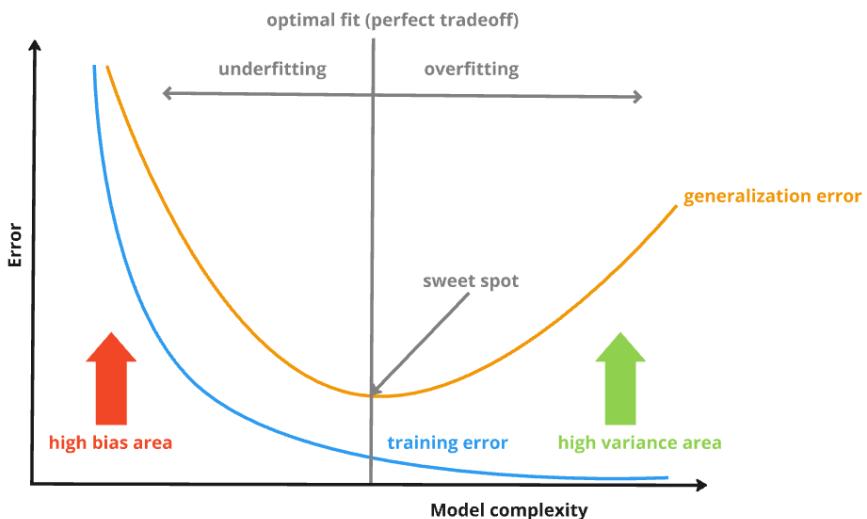


Figura 4.7.: Relación bias/variance con underfitting y overfitting en el contexto clásico. Al principio, el modelo muestra un alto sesgo, incapaz de capturar adecuadamente las relaciones entre los datos. A medida que aumenta su complejidad, tanto el rendimiento en entrenamiento como en test mejora, alcanzando 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. Imagen original del autor.

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.

4.4. Underfitting y overfitting

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 224 publicaciones en Scopus¹. Esta reciente relevancia ilustra el carácter **innovador** y **pionero** de este proyecto.

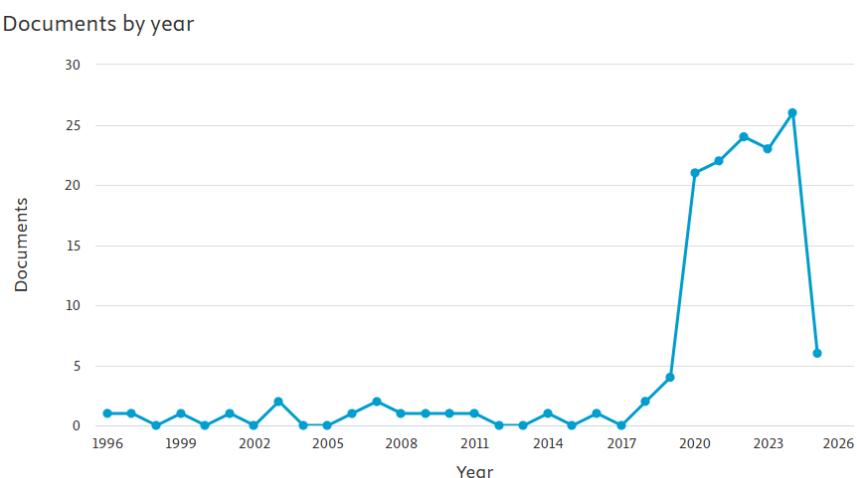


Figura 5.1.: Número de publicaciones relativas al Deep Double Descent en función del año de publicación.

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.

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

¹Encontradas 224 publicaciones a fecha 21 de marzo 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")).

y dejando de lado el análisis teórico intrínseco, limitando la comprensión completa de ciertos aspectos y resultados.

5.1. Origen y primeras manifestaciones

La primera publicación formal sobre el *Deep Double Descent* data de 1989, como se muestra en la Figura 5.1. En ese año, Vallet et al. [VCR89] presentaron, de manera empírica, este comportamiento en el ámbito de la física teórica, 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 el suceso 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 [WH60], y, más tarde, en su revisión del artículo en 2001 “Learning to Generalize” [Opp01].

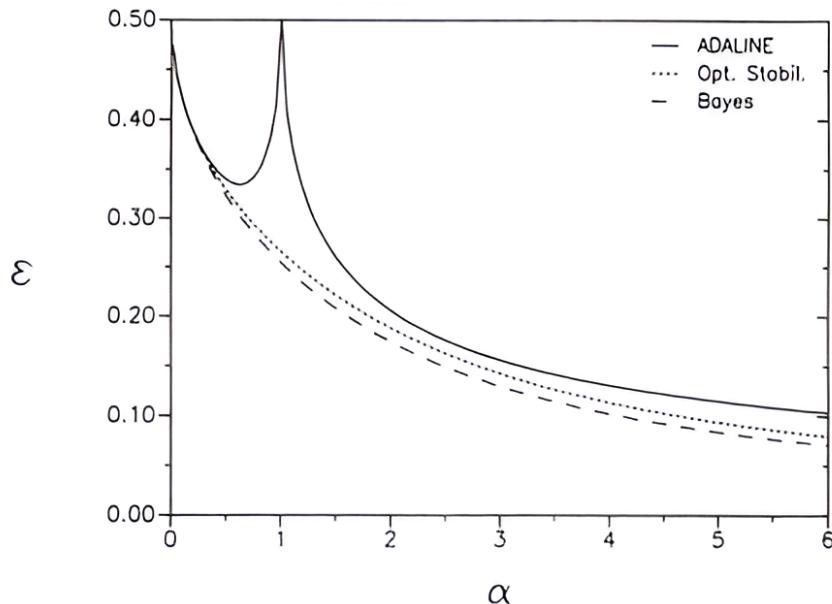


Figura 5.2.: Comparación del error de generalización (ϵ) para distintos modelos en función de la fracción entre el número de ejemplos aprendido y el número de parámetros (α). Se observa la curva del doble descenso para el modelo ADALINE [Opp95].

En sus representaciones (véase Figura 5.2), el error de generalización (ϵ) 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 α se approxima a 1, es decir, cuando el número de ejemplos de entrenamiento es similar al número de parámetros del modelo. En adición, demuestra que, para ciertas configuraciones, cuando N tiende

5. Trabajos relacionados

a infinito, la solución que proporciona la pseudoinversa mejora a medida que nos alejamos del “pico” ($\alpha = 1$).

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⁺19, GSd⁺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 doble descenso 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 prueba.

No obstante, fue Duin en el año 2000 [Du00] (véanse Figuras 6 y 7) 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⁺19, HCB⁺19, SLJ⁺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. En particular, se enlaza con investigaciones previas, como la de Zhang et al. en 2021 [ZBH⁺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).

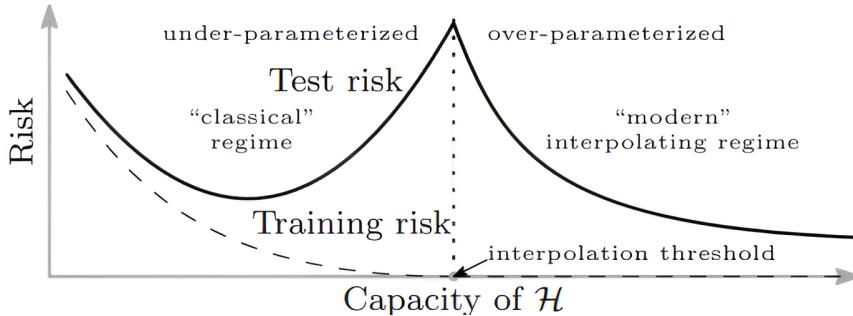


Figura 5.3.: Curvas para el error de entrenamiento (línea discontinua) y el error de generalización (línea continua) [BHMM19]. Antes del umbral de interpolación, se muestra la curva clásica en “U”. Despues de dicho umbral, se observa la curva del error moderna, induciendo el doble descenso.

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⁺24].

Posteriormente, Nakkiran et al. en 2019 [NKB⁺19] observaron que el doble descenso no 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 conjecturaron 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.

5.3. Avances recientes

En los últimos años, la comprensión del *Deep Double Descent* ha avanzado significativamente, con 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⁺21, CJvdS23]. Por otro lado, Schaeffer et al. en 2023 [SKR⁺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 pre-

5. Trabajos relacionados

sencia. 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. Asimismo, 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. Por último, Somepalli et al. en 2022 [SFB⁺22] investigaron la reproducibilidad del aprendizaje en redes neuronales y su relación con el *Deep Double Descent*.

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*. 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.

Para concluir este capítulo, se presenta en la Tabla 5.1 un resumen de las principales publicaciones relacionadas con el doble descenso, aunque las primeras manifestaciones no utilizan la denominación actual, junto con sus contribuciones más relevantes. Asimismo, se incluye en la tabla el número de citas de cada publicación, obtenido a partir de *Google Scholar*².

²Número de citas encontradas a día 14 de marzo de 2025.

Año	Referencia	Principales contribuciones	Citas
1989	Vallet et al. [VCR89]	Primera manifestación empírica del suceso sobre datos artificiales en la física teórica.	71
1991	Krogh & Hertz [KH91]	Evidenciaron, parcialmente, el fenómeno en el contexto de la regresión lineal.	2654
1995	Manfred Opper [Opp95]	Primeros resultados teóricos formales sobre datos artificiales.	71
2000	Robert Duin [Duioo]	Primeras evidencias empíricas usando datos del mundo real.	209
2019	Belkin et al. [BHMM19]	Unificaron la sabiduría clásica con los enfoques modernos y acuñaron el nombre de <i>Deep Double Descent</i> .	2331
2019	Nakkiran et al. [NKB ⁺ 19]	Introdujeron la complejidad efectiva del modelo y los distintos tipos de doble descenso.	1174
2021	Chen et al. [CMBK21]	Demostraron que el error puede presentar un número arbitrario de "picos" y que pueden controlarse.	80
2023	Schaeffer et al. [SKR ⁺ 23]	Analizaron los factores que provocan su aparición en modelos de regresión polinómica.	29

Tabla 5.1.: Resumen de los principales artículos junto con sus contribuciones.

Parte III.

Desarrollo Teórico y Empírico

6. Análisis teórico del Deep Double Descent

En este capítulo, nos vamos a encargar de abordar, de manera teórica, el doble descenso profundo. 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 logran 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 doble descenso profundo y unificar la sabiduría clásica con la práctica moderna, introducimos una nueva medida de capacidad del modelo, propuesta por Nakkiran et al. en [NKB⁺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 $\epsilon > 0$:

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

donde $L(g)$ 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 4.1.1.1) y un pequeño $\epsilon > 0$, si consideramos la tarea de predecir etiquetas basadas en n muestras aleatorias e independientes de $P[X, Y]$, entonces:

- **Región infraparametrizada.** Si $EMC_{P,\epsilon}(\mathcal{A})$ es suficientemente menor que n , entonces cualquier perturbación de \mathcal{A} que aumente su complejidad efectiva, disminuirá su error de generalización.
- **Región crítica.** Si $EMC_{P,\epsilon}(\mathcal{A}) \approx n$, entonces cualquier perturbación de \mathcal{A} que aumente su complejidad efectiva puede aumentar o disminuir su error de generalización.
- **Región sobreparametrizada.** Si $EMC_{P,\epsilon}(\mathcal{A})$ 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 ϵ , ni de una especificación formal para “suficientemente pequeño” y “suficientemente grande”. La hipótesis sugiere que hay un intervalo crítico alrededor del umbral de interpolación ($EMC_{P,\epsilon}(\mathcal{A}) = n$) 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.

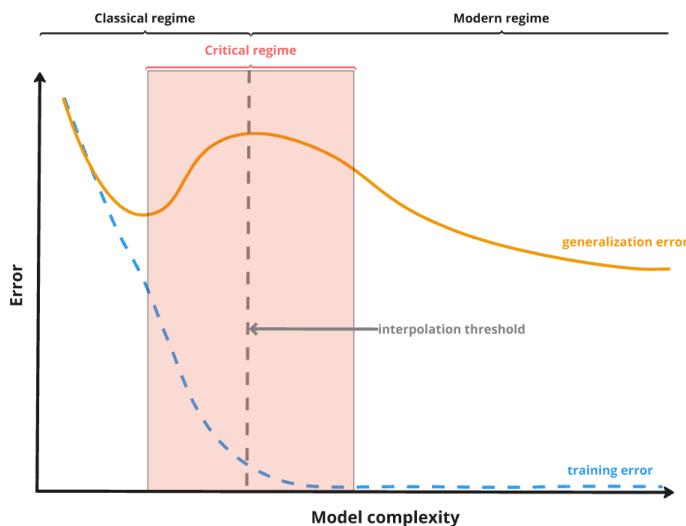


Figura 6.1.: Ejemplo de doble descenso con las distintas zonas de parametrización. Se puede observar la tradicional forma de “U” en la zona clásica, seguida de la zona crítica, donde el error de generalización es incierto, ya que puede aumentar o disminuir, y, finalmente, la zona moderna, donde el error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

6. Análisis teórico del Deep Double Descent

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 denominaremos por **doble descenso profundo**, o simplemente **doble descenso**.

De esta manera, 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 $L(g) = 0$. 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:

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

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 doble descenso, consideramos un problema de regresión lineal (basado en [SKR⁺23]) 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 ante un problema de regresión lineal en el que disponemos de N ejemplos, donde cada ejemplo está formado por los respectivos datos de entrada 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 $y \in \mathbb{R}$ 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 $(X^T X)^{-1} X^T$. 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, ya que habría menos restricciones que parámetros. Por tanto, debemos de 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:

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

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 conociendo que la matriz X tiene rango completo en esta región y cuenta con filas linealmente independientes. Así, la pseudoinversa viene dada por $X^T (X X^T)^{-1}$ (véase 2.2). De esta forma, el vector de pesos w puede ser expresado como:

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

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_{test} , dependiendo de la zona de parametrización en la que se encuentre:

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

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

6. Análisis teórico del Deep Double Descent

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:

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

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

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

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

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 prueba 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 ligadas 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 balanceo está correlacionado con los objetivos de regresión.

Por tanto, esos 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

6. Análisis teórico del Deep Double Descent

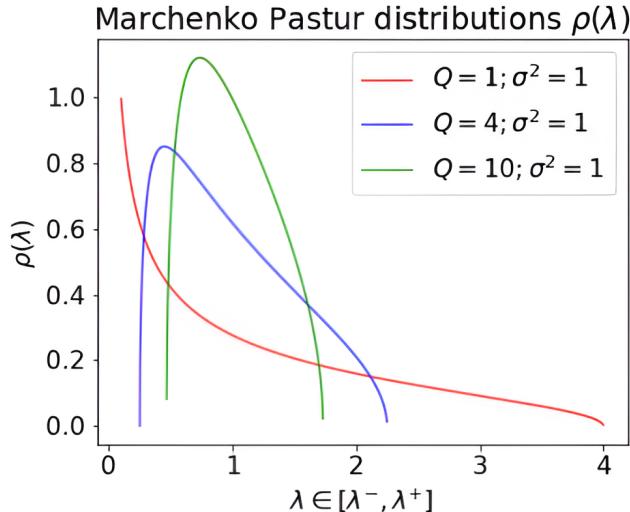


Figura 6.2.: Ejemplos de distribuciones de Marchenko-Pastur [MM18] para distintos valores de la relación de aspecto de la matriz ($Q = \frac{N}{P}$) y varianza fija. Cuando $Q = 1$, la mayoría de valores propios se acumulan alrededor del cero. A medida que Q aumenta, los valores propios mínimos se alejan progresivamente de cero.

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 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 6.2), 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 muestrean 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 covariaciones se hace más evidente 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. Sesgo inductivo 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 $x \in \mathbb{R}^D$ y $y \in \mathbb{R}$. Así, nuestro conjunto de entrenamiento presenta la siguiente forma:

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

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 η , 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: $Xw - y = Xw - XX^\dagger y + XX^\dagger y - y = Xw - XX^\dagger y - (I_D - XX^\dagger)y$.

6. Análisis teórico del Deep Double Descent

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 $\text{Im}(X)$ y $(I_D - XX^\dagger)y$ es la proyección en el complemento ortogonal de $\text{Im}(X)$, 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 qué tan 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.*

6.2. Sesgo inductivo del descenso de gradiente

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:

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

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 .

6. Análisis teórico del Deep Double Descent

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 Σ_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:

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

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_∞ 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):

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

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⁺24]. Cabe destacar que, por simplicidad, consideramos un problema de clasificación binaria con un dataset 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 multiclas [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. ℓ 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. ℓ es una función β -suave, es decir, el gradiente de ℓ es β -Lipschitz ($\forall u, v \in \mathbb{R}, \|\nabla \ell(u) - \nabla \ell(v)\| \leq \beta \|u - v\|$).
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

6. Análisis teórico del Deep Double Descent

implican que $\mathcal{L}(w)$ es una función $\beta\sigma_{\max}^2(X)$ -suave (ligado al hecho de que ℓ es β -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 $x_i \in \mathbb{R}^D$ y $y_i \in \{-1, 1\}$, donde $i \in \{1, \dots, N\}$ y nos centramos en minimizar la función de pérdida dada por:

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

donde $w \in \mathbb{R}^D$ es el vector de parámetros y ℓ 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 η :

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

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 β -suave no negativa. Si $\eta < \frac{2}{\beta}$, entonces, para cualquier w_0 y utilizando el método del descenso de gradiente dado por:*

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

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.

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. $\forall i \in \{1, \dots, N\} : \lim_{t \rightarrow \infty} y_i w^{(t)T} x_i = \infty$.

6.3. Optimización en la zona sobreparametrizada

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 ℓ . Por lo 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 como se indica en [HTFor1], que afirma: “Un modelo con error de entrenamiento cero sobreajusta los datos de entrenamiento y, por lo general, generalizará mal”, refleja cómo las soluciones de los sistemas sobreparametrizados pueden mejorar la generalización.

Como hemos observado en secciones anteriores, la sobreparametrización resulta beneficiosa dentro del marco del aprendizaje estadístico. Este comportamiento, aunque contraintuitivo [HTFor1] “Un modelo con cero error de entrenamiento está por encima de los datos de entrenamiento y, por lo general, generalizará mal”, 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 conve-

6. Análisis teórico del Deep Double Descent

xidad de ℓ). 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 vecindario alrededor de ningún minimizador global en el que el paisaje de la función de pérdida sea convexo. Esto contrasta con lo que ocurre en los sistemas infraparametrizados, donde dicho vecindario suele existir, aunque sea extremadamente pequeño (véase Figura 6.3).

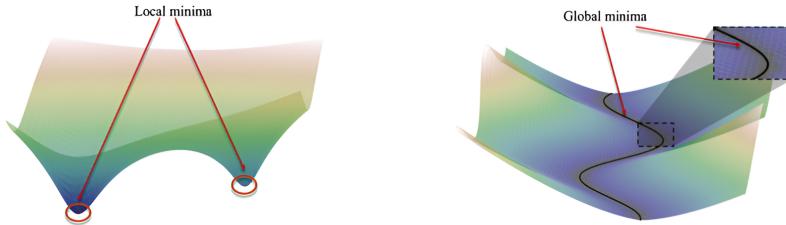


Figura 6.3.: Ejemplos de paisajes de la función de pérdida en redes neuronales [LZB21]. A la izquierda, se muestra un paisaje localmente convexo alrededor de los mínimos locales, ligado a la zona infraparametrizada. A la derecha, se muestra un paisaje incompatible con la convexidad local con múltiples mínimos globales, 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 dataset \mathcal{D} compuesto por N -pares de ejemplos de entrenamiento, 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 ecuaciones¹. Agregándolas todas en una única función, podemos escribir:

$$\mathcal{F}(w) = y, \text{ donde } w \in \mathbb{R}^P, y \in \mathbb{R}^N, \mathcal{F}(\cdot) : \mathbb{R}^P \rightarrow \mathbb{R}^N. \quad (6.4)$$

De esta manera, $(\mathcal{F}(w)) = f(w; x_i)$. Una solución exacta de la Ecuación (6.4) corresponde a la interpolación.

¹Para 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.

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, 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 ninguna vecindad de w^* .*

Demostración. Se presenta una idea intuitiva de cómo realizar la demostración del resultado anterior. Para ello, tomaremos dos puntos distintos del vecindario de la solución w^* y evaluaremos las matrices hessianas de la función de pérdida en dichos puntos. Una vez evaluadas, podemos verificar que alguna de las matrices anteriores es no negativa, 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 esa vecindad, lo cual es una condición necesaria para que la función sea localmente convexa.

Por último, se referencia al lector más curioso al Apéndice B para la demostración detallada de la proposición. □

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 la vecindad 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 6.3 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 vecindario 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 μ -PL). Decimos que una función no negativa \mathcal{L} verifica la condición μ -PL (Polyak-Lojasiewicz) en un conjunto $\mathcal{S} \subset \mathbb{R}^P$ para $\mu > 0$, si

6. Análisis teórico del Deep Double Descent

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

Teorema 6.11. Supongamos que la función de pérdida $\mathcal{L}(w)$ es β -suave y cumple la condición μ -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 $\eta \leq \frac{1}{\sup_{w \in B(w_0, R)} \|\mathcal{H}_{\mathcal{L}}(w)\|}$ converge a una solución global en $B(w_0, R)$.

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 $R = \frac{2\sqrt{2\beta\mathcal{L}(w_0)}}{\mu}$.

En el caso inicial, cuando $t = 0$, es trivial que $w_0 \in B(w_0, R)$. Supongamos que, por inducción, para un $t \geq 0$, w_t está en la bola $B(w_0, R)$. Para probar que $w_{t+1} \in B(w_0, R)$, usando que la función de pérdida \mathcal{L} es β -suave, tenemos que:

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

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

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 μ -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

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

donde $K(w) \in \mathcal{M}_{N \times P}(\mathbb{R})$ es una matriz semidefinida positiva.

Definición 6.13 (Condicionamiento uniforme). Decimos que $\mathcal{F}(w)$ está μ -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 μ -PL.

Teorema 6.14 (Condicionamiento uniforme \implies Condición μ -PL). Si $\mathcal{F}(w)$ está μ -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 μ -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 μ -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

$$\mathcal{S}_{\text{sing}} = \{w \in \mathbb{R}^P \mid \lambda_{\min}(K(w)) = 0\} = \{w \in \mathbb{R}^P \mid \text{rango } \mathcal{D}\mathcal{F}(w) < n\}.$$

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 μ , es probable encontrar puntos que no satisfacen la condición μ -PL alrededor de $\mathcal{S}_{\text{sing}}$, el cual es un subconjunto de



Figura 6.4.: Ejemplo de función de pérdida que satisface la condición μ -PL [LZB21]. La función de pérdida es μ -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 μ -PL se interseca con el conjunto de mínimos globales de la función de pérdida.

baja dimensión en \mathbb{R}^P . Esto puede observarse en la Figura 6.4.

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 μ -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 μ -PL.

6.3.1. Elección de la mejor 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, es decir, cualquier hipótesis $g \in S \subset \mathcal{H}$.

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 6.5, donde, una vez que tenemos suficientes parámetros para ajustar todos los datos de entrenamiento, empezamos a obtener distintos modelos que son minimizadores globales.

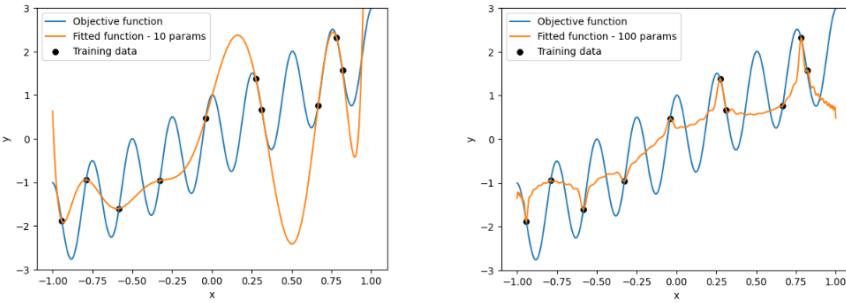


Figura 6.5.: Distintos modelos polinómicos (línea naranja) para aproximar una función objetivo (línea azul). Ambos modelos se ajustan de manera perfecta a los datos de entrenamiento (puntos azules); sin embargo, el segundo modelo generaliza mejor al estar regularizado hacia una solución de norma pequeña. Imagen del autor inspirada en [SKR⁺23].

Extrayendo conclusiones de la Figura 6.5 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 minimizadores 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 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 elegida de S_2 sea más suave,

6. Análisis teórico del Deep Double Descent

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.

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”.

6.4. Resto de desarrollos a realizar

6.5. Aproximación no lineal

6.5.1. Analogía con el deep double descent

6.6. Conclusión

	Régimen clásico (infraparametrizado)	Régimen moderno (sobreparametrizado)
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 $\mu\text{-PL}$
Modelo óptimo	Sweet spot (mínimo de la curva “U”) (difícil de conseguir)	Cualquier minimizador global (fácil de encontrar)
Convergencia del GD	GD converge al mínimo local	GD converge a un mínimo global

Tabla 6.1.

7. Análisis empírico del Deep Double Descent

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 datasets utilizados en este estudio: MNIST, CIFAR-10, CIFAR-100, Flowers102.

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) [LC05] es un conjunto de datos de imágenes ampliamente utilizado para tareas de clasificación. 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 tiene un tamaño de 28×28 píxeles.

7.1.1.2. CIFAR-10 y CIFAR-100

CIFAR-10 [KNH] 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 32×32 píxeles y está en formato RGB, es decir, cada imagen cuenta con 3 capas, cada una asociada con un color del formato RGB.

Por otro lado, CIFAR-100 [KNH] es un conjunto de datos similar a CIFAR-10, 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 32×32 píxeles en formato RGB.

7. Análisis empírico del Deep Double Descent

7.1.1.3. Flowers102

El dataset Flowers102 [NZo8] es un conjunto de datos utilizado comúnmente en tareas de clasificación de imágenes, especialmente en el reconocimiento de imágenes de flores. Este dataset contiene 102 categorías diferentes de flores, con un total de 8189 imágenes de resolución variable en formato RGB, aunque la mayoría tienen un tamaño de 224×224 píxeles. Cada categoría de flores incluye entre 40 y 258 imágenes, lo que permite que el dataset tenga diversidad de muestras dentro de cada clase.

La distribución de las imágenes dentro del dataset es la siguiente: 1020 imágenes se utilizan para entrenamiento, 1020 imágenes para validación y 6149 imágenes para el conjunto de test. La gran diferencia en el número de imágenes entre los conjuntos de entrenamiento/validación y el conjunto de test hace que este dataset sea particularmente complicado de predecir, dado que las condiciones de las imágenes (ángulos, fondos y luces) varían considerablemente entre las clases.

Para asegurar que todas las imágenes de este dataset tengan las mismas dimensiones y reducir la complejidad computacional, las imágenes se escalan a una resolución de 32×32 píxeles en formato RGB. Al reducir la resolución de las imágenes, se facilita el entrenamiento de los modelos, aunque se podría comprometer significativamente su capacidad para aprender características relevantes para la clasificación.

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 de los datasets, 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 facilita la comparación entre ellos.

Dataset	Nº imágenes	Tamaño imagen	Clases	Entrada
MNIST	70000	28×28 píxeles (escala de grises)	10	784
CIFAR-10	60000	32×32 píxeles (RGB)	10	3072
CIFAR-100	60000	32×32 píxeles (RGB)	100	3072
Flowers102	8189	32×32 píxeles (RGB)	102	3072

Tabla 7.1.: Resumen de los datasets utilizados. En la tabla aparecen las principales características de cada dataset, donde la última columna 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 datasets en los experimentos. En particular, cuando se haga uso de un subconjunto, se empleará la siguiente notación:

$$\text{dataset[train/test]}$$

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 prueba. 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. Arquitecturas utilizadas

En esta sección se presentan las arquitecturas diseñadas y utilizadas con el objetivo de 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.

7.1.2.1. 2NN

Como primer modelo se propone una arquitectura muy simple, formada únicamente por dos capas densas. Para ello, primeramente 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 conectarán las neuronas de salida de la primera capa densa con un número de neuronas que será igual al número de clases del datasets con el que estemos trabajando.

El diseño y elección de este modelo se basa en su relativa sencillez, que será de gran utilidad a la hora de realizar diversidad de experimentos, lo que lo convierte en una opción ideal para llevar a cabo una amplia variedad de experimentos al permitir tiempos de entrenamiento significativamente más bajos en comparación con otras arquitecturas utilizadas.

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).

Arquitectura	Entrada	Número de parámetros
2NN	Imagen de tamaño $n \times n \times l$	$(n^2 \times l) \times k + k + c \times k + c$

Tabla 7.2.: Resumen de las arquitecturas 2NN. En la tabla 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.

7. Análisis empírico del Deep Double Descent

7.1.2.2. ResNet-18 modificada

ResNet-18 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. Para solucionarlo, ResNet introduce el concepto de las conexiones residuales, que permiten que el flujo de información pase a través de capas sin ser afectado por los gradientes de las capas anteriores.

ResNet-18, 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 7×7 , 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 capa convolucional. Finalmente, aparece una capa densa que será la salida de la red (véase Figura 7.1 para más detalle).

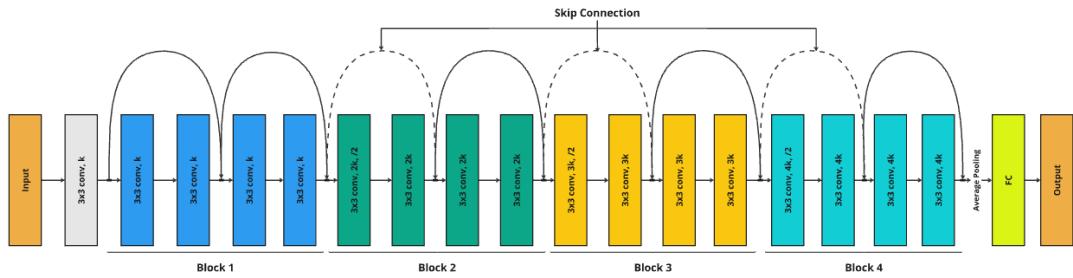


Figura 7.1.: Arquitectura ResNet18 modificada. Podemos observar los 4 bloques convolucionales (distinguidos por color), cada uno de ellos con su correspondiente número de filtros ($[k, 2k, 3k, 4k]$ con $k \in \mathbb{N}$), junto con las conexiones residuales. Imagen original del autor.

La arquitectura que usaremos durante el transcurso de los experimentos 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 poder modificar el número de parámetros de la red y crear “distintas” arquitecturas.

Para ello, usaremos como número de filtros para cada bloque convolucional la secuencia: $[k, 2k, 3k, 4k]$ [NKB⁺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 ResNet-18 original.

Finalmente y dado que el número de parámetros asociados a esta arquitectura es tedioso de indicar (debido a la gran cantidad de capas que la conforman), se expone una tabla con los principales valores del parámetro k (número de filtros) utilizados en el desarrollo de la

práctica (véase Tabla 7.3).

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 7.3.: Número de parámetros de las arquitecturas ResNet18 modificadas. Se muestra el número de parámetros dada un determinado valor k y para 10 clases de salida.

7.1.2.3. 3CNN

Siguiendo la misma idea de la subsección anterior, en la que se varía el número de filtros en cada capa convolucional, se propone una nueva arquitectura 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 los dos modelos expuestos previamente, 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 7.2).

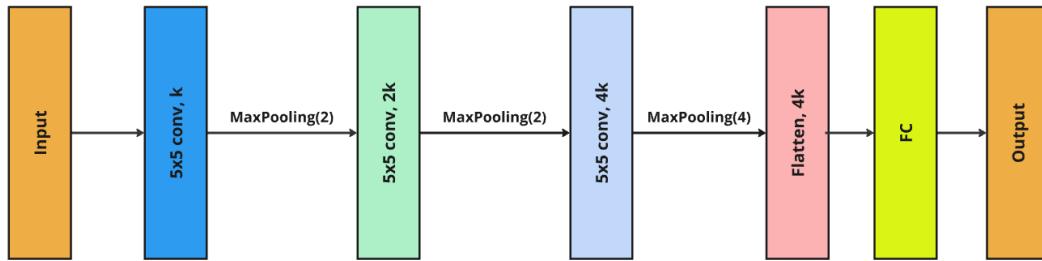


Figura 7.2.: Arquitectura 3CNN. Podemos observar los 3 bloques convolucionales (distinguidos por color), cada uno de ellos con su correspondiente número de filtros ($[k, 2k, 4k]$ con $k \in \mathbb{N}$). Imagen original del autor.

Finalmente, la Tabla 7.4 muestra el número de parámetros para algunos valores del parámetro k , lo que permite realizar una comparación con la arquitectura ResNet anterior.

Valor de k	Número de parámetros
20	$\approx 103 \text{ K}$
45	$\approx 512 \text{ K}$
64	$\approx 1,03 \text{ M}$

Tabla 7.4.: Número de parámetros de las 3CNN. Se muestra el número de parámetros dada un determinado valor k y para 10 clases de salida.

7. Análisis empírico del Deep Double Descent

7.1.3. 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 [PGM⁺19] 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 del lote (*batch size*), variará entre 128 y 256, de manera similar a la literatura existente, lo que representa un valor relativamente alto, con el objetivo de acelerar el proceso de entrenamiento. Respecto a la tasa de aprendizaje del optimizador utilizado (Adam en nuestro caso), se empleará la 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 C, 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.

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 este suceso 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, 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.

7.2.1. Aproximación polinómica de Legendre

Este primer 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 en los experimentos posteriores.

Para ello, se busca replicar el resultado de la Imagen 2 de [SKR⁺23] (véase Figura 7.4). El experimento trata de aproximar una función objetivo ($y(x) = 2x + \cos(25x)$) mediante el uso de la aproximación polinómica de Legendre para distintos grados de dicho polinomio y sobre un conjunto de 10 puntos obtenidos al muestrear la función anterior. Este experimento no utilizará redes neuronales, sino que se basa en una simple regresión polinómica.

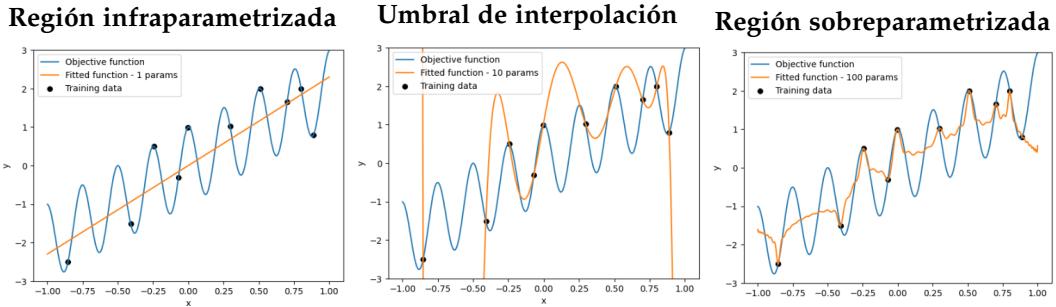


Figura 7.3.: Intuición del *Deep Double Descent* usando regresión polinómica. Cuando nos encontramos en la región infraparametrizada, el modelo no es capaz de capturar todos los datos de entrenamiento, haciendo que el bias del modelo sea grande, aunque la varianza sea pequeña. En el umbral de interpolación, 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.

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 solución única. En este caso, el grado del polinomio coincide con el número de puntos, lo que no asegura garantizar una buena generalización debido a la “rigidez” del modelo al presentar numerosos “picos” (véase Figura 7.3).

Al superar dicho umbral, el modelo adquiere mayor flexibilidad para aproximar los datos, lo que da lugar a una función cada vez más “suave”. Además, el aumento en el número de parámetros más allá de dicho umbral permite la existencia de múltiples modelos de interpolación para cada grado, facilitando la selección de una opción que generalice bien. Esto puede observarse en la Figura 7.4.

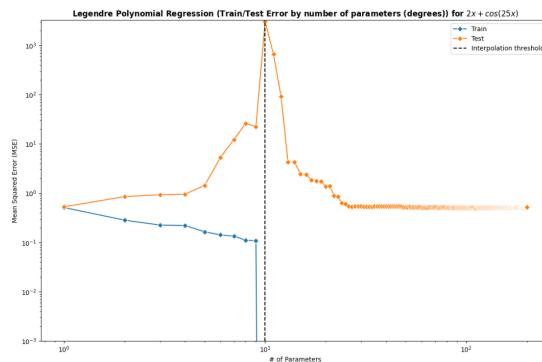


Figura 7.4.: Doble descenso al utilizar aproximación polinómica de Legendre, donde el umbral de interpolación corresponde al número de datos de entrenamiento.

7. Análisis empírico del Deep Double Descent

7.2.1.1. Función cuadrática

7.2.1.2. Función exponencial

7.2.1.3. Función hiperbólica

7.2.2. 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 en las etiquetas. De este modo, añadimos este tipo de doble descenso a las variantes clásicas propuestas por Nakkiran et al. en [NKB⁺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 y entrenando cada modelo durante 1000 épocas.



Figura 7.5.: 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 doble descenso 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.

En la Figura 7.5 se observa que 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 doble descenso, el pico del error de test aumenta al incrementar el porcentaje de ruido, ya que el modelo, eventualmente, memorizará dicho ruido, lo cual concuerda con lo planteado en la literatura científica.

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 7.5) 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 7.6 podemos verificar cómo, al aumentar el porcentaje de ruido en las etiquetas, el umbral de interpolación se desplaza hacia la derecha.

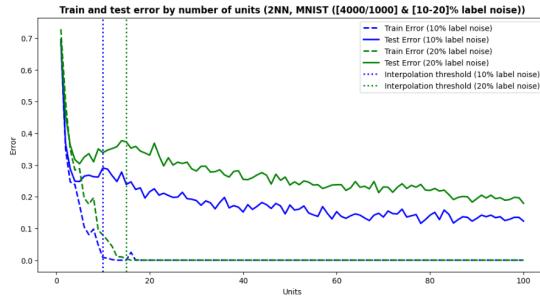


Figura 7.6.: Error en entrenamiento, test y umbral de interpolación respecto a la arquitectura 2NN sobre el subconjunto MNIST[4000/1000] con distinto nivel de ruido añadido en las etiquetas.

Modelo	Dataset	Ruido en etiquetas	Entrenamiento
2-NN (1 – 100)	MNIST[4000/1000]	×	26h 10min
2-NN (1 – 100)	MNIST[4000/1000]	10 %	26h 50min
2-NN (1 – 100)	MNIST[4000/1000]	20 %	27h 23min
2-NN (1 – 100)	MNIST[4000/1000]	30 %	28h 8min
2-NN (1 – 100)	MNIST[4000/1000]	40 %	30h 14min

Tabla 7.5.: Resumen de los experimentos para el doble descenso por nivel de ruido.

7.2.3. Sample-wise double descent

En esta subsección se presentan los experimentos realizados para analizar el *sample-wise double descent*, el cual se manifiesta al modificar la cantidad de datos empleados durante el entrenamiento de un modelo específico. Este comportamiento destaca una zona de interés donde, al comparar las curvas del error de prueba, se observa que entrenar con un mayor número de ejemplos puede, de manera casi paradójica (en el sentido que, generalmente, en el aprendizaje automático siempre se busca tener la mayor cantidad de datos posible de cara a entrenar), empeorar el rendimiento del modelo.

Dado que la experimentación presentada en la literatura existente resulta excesivamente costosa de replicar, se han diseñado experimentos más accesibles que permiten analizarlo manteniendo un enfoque lo suficientemente 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 el número de ejemplos de entrenamiento, el pico del error de test se desplazará hacia la derecha (véase la imagen izquierda de la Figura 7.7), creando una región, 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 7.7). Sin embargo, es importante resaltar que, fuera de esta zona específica, el entrenamiento con un mayor número de ejemplos suele ser beneficioso para el rendimiento general del modelo, es decir, el modelo debería obtener menor error de test (el área bajo la curva de error fuera de dicha

7. Análisis empírico del Deep Double Descent

región debe ser menor para el modelo entrenado con un mayor número de ejemplos).

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 y entrenaremos cada modelo durante 1000 épocas. 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 agregarán un ruido del 10 % en sus etiquetas.

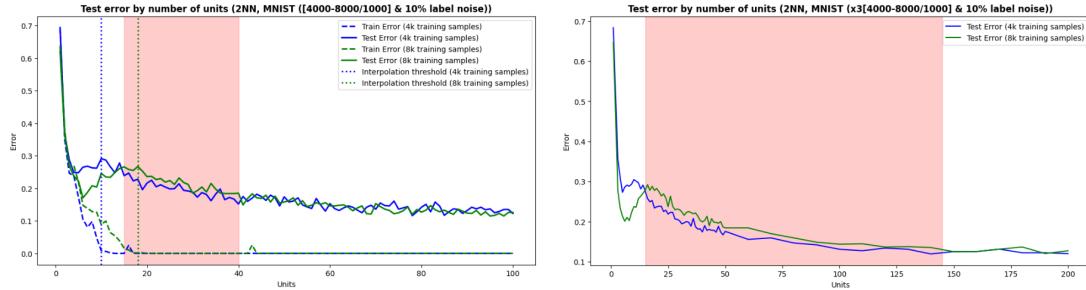


Figura 7.7.: Ejemplo de *sample-wise double descent* para la red 2NN sobre dos subconjuntos de MNIST, utilizando 4000 y 8000 ejemplos de entrenamiento. A la izquierda aparece el resultado de una única realización del experimento. A la derecha aparece el resultado obtenido al realizar la media de 3 experimentos.

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 la imagen izquierda de la Figura 7.7). Es decir, el modelo requiere de un mayor número de parámetros para memorizar un mayor número de ejemplos de entrenamiento, lo que, a priori, parece tener sentido con la complejidad efectiva del modelo.

Finalmente, se muestra en la Tabla 7.6 el tiempo de ejecución necesario para la realización de este experimento. Cabe destacar que, en el experimento donde se calcula la media de 3 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
2-NN (1 – 100)	MNIST[4000/1000]	26h 50min
2-NN (1 – 100)	MNIST[8000/1000]	59h
2-NN (1 – 200) × 3	MNIST[4000/1000]	53h 10min
2-NN (1 – 200) × 3	MNIST[8000/1000]	104h 30min

Tabla 7.6.: Resumen de los experimentos para el doble descenso por número de ejemplos de entrenamiento.

7.3. Conclusión

Parte IV.

Conclusiones y Trabajos Futuros

8. Conclusiones

9. Trabajos futuros

A. Apéndice

En este apéndice se muestra la demostración del Lema 6.7, extraída de [SHN⁺ 24].

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

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

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. Usando una propiedad conocida de las funciones β -suaves:

$$|f(x) - f(y) - \nabla f(y)^T(x - y)| \leq \|x - y\|^2.$$

Dado que la función $\mathcal{L}(w)$ es β -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

$$\sum_{u=0}^t \|\nabla \mathcal{L}(w^{(u)})\|^2 \leq \sum_{u=0}^t \frac{\mathcal{L}(w^{(u)}) - \mathcal{L}(w^{(u+1)})}{\eta \left(1 - \frac{\beta \eta}{2}\right)} = \frac{\mathcal{L}(w_0) - \mathcal{L}(w^{(\tau+1)})}{\eta \left(1 - \frac{\beta \eta}{2}\right)}.$$

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, esto implica que

$$\sum_{u=0}^{\infty} \|\nabla \mathcal{L}(w^{(u)})\|^2 < \infty,$$

y, finalmente, nos queda el resultado buscado:

$$\|\nabla \mathcal{L}(w^{(\tau)})\|^2 \rightarrow 0.$$

A. Apéndice

□

B. Apéndice

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 ninguna vecindad de w^* .

Demostración. La matriz Hessiana de una función de pérdida $\mathcal{L}(\mathcal{F}(w))$ viene dada por:

$$H_{\mathcal{L}}(w) = D\mathcal{F}(w)^T \frac{\partial^2 \mathcal{L}}{\partial \mathcal{F}^2}(w) D\mathcal{F}(w) + \sum_{i=1}^n \frac{\partial \mathcal{L}}{\partial \mathcal{F}_i}(w) H_{\mathcal{F}_i}(w).$$

Consideramos ahora las matrices Hessianas de dos puntos $w^* + \delta$ y $w^* - \delta$ ($\delta \in \mathbb{R}^P$) que están en una vecindad suficientemente pequeña de w^* . La Hessiana de la función de pérdida en estos dos puntos es:

$$\begin{aligned} H_{\mathcal{L}}(w^* + \delta) &= \underbrace{D\mathcal{F}(w^* + \delta)^T \frac{\partial^2 \mathcal{L}}{\partial \mathcal{F}^2}(w^* + \delta) D\mathcal{F}(w^* + \delta)}_{A(w^* + \delta)} \\ &\quad + \sum_{i=1}^n \left(\frac{d}{dw} \left(\frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) \delta \right)_i H_{\mathcal{F}_i}(w^* + \delta) + o(\|\delta\|), \\ H_{\mathcal{L}}(w^* - \delta) &= \underbrace{D\mathcal{F}(w^* - \delta)^T \frac{\partial^2 \mathcal{L}}{\partial \mathcal{F}^2}(w^* - \delta) D\mathcal{F}(w^* - \delta)}_{A(w^* - \delta)} \\ &\quad - \sum_{i=1}^n \left(\frac{d}{dw} \left(\frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) \delta \right)_i H_{\mathcal{F}_i}(w^* + \delta) + o(\|\delta\|). \end{aligned}$$

donde los términos $A(w^* + \delta)$ y $A(w^* - \delta)$ son matrices de rango a lo sumo n , ya que $D\mathcal{F}$ 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 δ 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:

$$v^T H_{\mathcal{F}_k}(w^* + \delta) v \neq 0, \quad v^T H_{\mathcal{F}_k}(w^* - \delta) v \neq 0.$$

B. Apéndice

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:

$$v^T H_{\mathcal{L}}(w^* + \delta)v = \sum_{i=1}^n \left(\frac{d}{dw} \left(\frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) \delta \right)_i v^T H_{\mathcal{F}_i}(w^* + \delta)v + o(\|\delta\|),$$

$$v^T H_{\mathcal{L}}(w^* - \delta)v = - \sum_{i=1}^n \left(\frac{d}{dw} \left(\frac{\partial \mathcal{L}}{\partial \mathcal{F}}(w^*) \right) \delta \right)_i v^T H_{\mathcal{F}_i}(w^* - \delta)v + o(\|\delta\|).$$

Para finalizar tenemos que mostrar que, para un δ 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 una vecindad cercana 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 δ 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 δ es lo suficientemente pequeño. A modo de conclusión, dado un δ 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 tiene ninguna vecindad convexa alrededor de w^* .

□

C. Apéndice

En este apéndice se presentan dos experimentos relacionados con el uso de diferentes tamaños de lote y tasas de aprendizaje para un modelo que exhibe el doble descenso. Para ello, se empleará la arquitectura 2NN sobre el subconjunto MNIST[4000/1000], al cual se le añadirá un porcentaje variable de ruido, y donde se utilizarán distintas tasas de aprendizaje con el optimizador Adam.

Double descent con distinto tamaño de batch

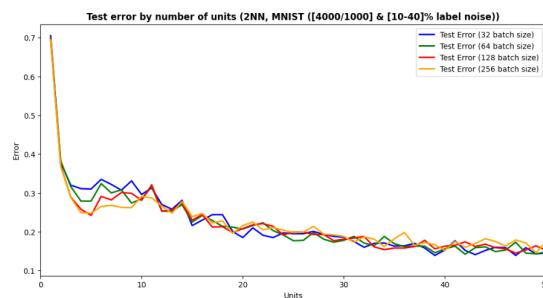


Figura C.1.: Error en test respecto a distintas configuraciones de tamaño de lote obtenido por la red 2NN sobre el subconjunto MNIST[4000/1000] con 10 % de ruido agregado.

En la Figura C.1 se observa que, para distintas configuraciones de tamaño de batch, se manifiesta el doble descenso, siendo más notable al utilizar tamaños de batch mayores. Por otro lado, la Tabla C.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 más adecuada para los distintos experimentos a realizar, ya que permite observar el fenómeno de forma más clara y con menor coste computacional.

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 C.1.: Resumen de los experimentos realizados para el doble descenso con distinto tamaño de lote.

C. Apéndice

Double descent con distinto learning rate

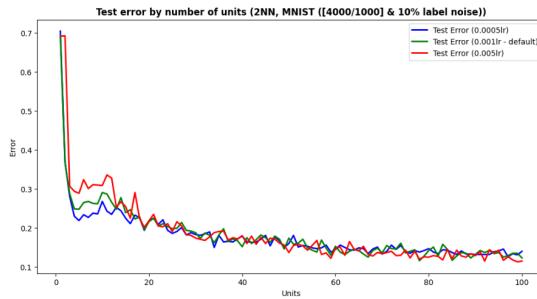


Figura C.2.: Error en test respecto a distintas configuraciones de tasa de aprendizaje obtenido por la red 2NN sobre el subconjunto MNIST[4000/1000] con 10 % de ruido agregado.

En la Figura C.2 podemos observar cómo se manifiesta el doble descenso para distintas configuraciones de la tasa de aprendizaje. En particular, el “pico” del error de test se hace más pronunciado a medida que la tasa de aprendizaje aumenta, dado que para una tasa igual a 5×10^{-4} , el máximo error en test es mayor que para el valor por defecto de la tasa de aprendizaje (10^{-4}), y este, a su vez, es mayor que para la mínima tasa de aprendizaje utilizada (5×10^{-5}).

En conclusión, esto nos indica que, si un modelo presenta el doble descenso, no es necesario preocuparse en exceso por ajustar de manera fina la tasa de aprendizaje, puesto que, dentro de ciertos rangos de valores para dicha tasa de aprendizaje, el modelo sigue manifestando el suceso, lo que indica que el uso por defecto de la tasa de aprendizaje resulta ser suficiente.

Finalmente, para concluir esta subsección se muestra en la Tabla C.2 un resumen de los experimentos realizados junto con sus respectivos tiempos de entrenamiento.

Modelo	Dataset	Tasa aprendizaje	Entrenamiento
2NN (1 – 100)	MNIST[4000/1000 – 10 % noise]	0.005	23h 53min
2NN (1 – 100)	MNIST[4000/1000 – 10 % noise]	0.001 (default)	26h 50min
2NN (1 – 100)	MNIST[4000/1000 – 10 % noise]	0.0005	27h 32min

Tabla C.2.: Resumen de los experimentos realizados para el doble descenso con distintas tasas de aprendizaje.

Glosario

\mathbb{R} Conjunto de números reales.

\mathbb{N} Conjunto de números naturales.

i.d.d. Independientes e idénticamente distribuidas.

$\mathbb{E}[X]$ Esperanza de la variable aleatoria X .

$P[X, Y]$ Probabilidad conjunta de las variables aleatorias X e Y .

SVD Descomposición en valores singulares.

X^\dagger Pseudoinversa de la matriz X .

Descenso de gradiente Método de optimización utilizado para minimizar una función de pérdida.

\mathcal{H} Conjunto de hipótesis de un modelo formado por las hipótesis candidatas y la función objetivo.

\mathcal{D} Conjunto de datos de entrenamiento.

E_{out} Error fuera de la muestra o error de generalización.

E_{in} Error dentro de la muestra o error de entrenamiento.

Complejidad efectiva del modelo Máximo número de ejemplos de entrenamiento en el que el modelo obtiene un E_{in} próximo a cero.

Bibliografía

- [AMMIL12] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, y Hsuan-Tien Lin. *Learning From Data*. AML-Book, 2012.
- [AS17] Madhu S. Advani y Andrew M. Saxe. High-dimensional dynamics of generalization error in neural networks, 2017.
- [BB23] Christopher Michael Bishop y Hugh Bishop. *Deep Learning - Foundations and Concepts*. 1 edición, 2023.
- [BD09] Shai Ben-David. *Theory-Practice Interplay in Machine Learning - Emerging Theoretical Challenges*, volumen 5781 de *Lecture Notes in Computer Science*. Springer, 2009.
- [BEHW87] Anselm BLUMER, Andrzej EHRENFEUCHT, David HAUSSLER, y Manfred K. WAR-MUTH. Occam's razor. 1987.
- [Bel21] Mikhail Belkin. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation, 2021.
- [BGKP22] Julius Berner, Philipp Grohs, Gitta Kutyniok, y Philipp Petersen. *The Modern Mathematics of Deep Learning*, páginas 1–111. Cambridge University Press, December 2022.
- [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*, 116(32):15849–15854, July 2019.
- [Bis95] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
- [Biso06] Christopher M Bishop. *Pattern Recognition and Machine Learning*, volumen 4 de *Information science and statistics*. Springer, 2006.
- [Blu21] Avrim Blum. Mathematical toolkit spring 2021, lecture 5, April 13 2021. Notes based on notes from Madhur Tulsiani.
- [Brb18] Randall Balestrierio y richard baraniuk. A spline theory of deep learning. En Jennifer Dy y Andreas Krause, editores, *Proceedings of the 35th International Conference on Machine Learning*, volumen 80 de *Proceedings of Machine Learning Research*, páginas 374–383. PMLR, 10–15 Jul 2018.
- [Bry95] Włodzimierz Bryc. *The Normal Distribution: Characterizations with Applications*. Springer New York, NY, 1995.
- [CJvdS23] Alicia Curth, Alan Jeffares, y Mihaela van der Schaar. A u-turn on double descent: Rethinking parameter counting in statistical learning, 2023.
- [CMBK21] Lin Chen, Yifei Min, Mikhail Belkin, y Amin Karbasi. Multiple descent: Design your own generalization curve, 2021.
- [CRF⁺25] Ben Cottier, Robi Rahman, Loredana Fattorini, Nestor Maslej, Tamay Besiroglu, y David Owen. The rising costs of training frontier ai models, 2025.
- [Dem14] Amir Dembo. *Probability Theory: STAT310/MATH230*. CreateSpace Independent Publishing Platform, 2014.
- [DLK23] Xander Davies, Lauro Langosco, y David Krueger. Unifying grokking and double descent, 2023.

Bibliografía

- [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, December 2021.
- [Duioo] R.P.W. Duin. Classifiers in almost empty spaces. En *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volumen 2, páginas 1–7 vol.2, 2000.
- [EEAMT22] Mohamed Elasri, Omar Elharrouss, Somaya Ali Al-Maadeed, y Hamid Tairi. Image generation: A review. *Neural Processing Letters*, 54:4609–4646, 2022.
- [FIS14] S.H. Friedberg, A.J. Insel, y L.E. Spence. *Linear Algebra*. Pearson Education, 2014.
- [GB10] Xavier Glorot y Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. En *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volumen 9 de *Proceedings of Machine Learning Research*, páginas 249–256. PMLR, 13–15 May 2010.
- [GBC16] Ian Goodfellow, Yoshua Bengio, y Aaron Courville. *Deep Learning*. MIT Press, 2016. Book in preparation for MIT Press.
- [GBD92] Stuart Geman, Elie Bienenstock, y René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.
- [GLSS19] Suriya Gunasekar, Jason Lee, Daniel Soudry, y Nathan Srebro. Implicit bias of gradient descent on linear convolutional networks, 2019.
- [GSd⁺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. *Phys. Rev. E*, 100:012115, Jul 2019.
- [HCB⁺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.
- [HT20] Fengxiang He y Dacheng Tao. Recent advances in deep learning theory. *ArXiv*, abs/2012.10931, 2020.
- [HTFo1] Trevor Hastie, Robert Tibshirani, y Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.
- [HY20] Reinhard Heckel y Fatih Furkan Yilmaz. Early stopping in deep networks: Double descent and how to eliminate it, 2020.
- [HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, y Jian Sun. Deep residual learning for image recognition, 2015.
- [KH91] Anders Krogh y John A. Hertz. A simple weight decay can improve generalization. En *Proceedings of the 5th International Conference on Neural Information Processing Systems, NIPS’91*, páginas 950–957, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.
- [KLW19] Uday Kamath, John Liu, y James Whitaker. *Deep Learning for NLP and Speech Recognition*. 01 2019.
- [KNH] Alex Krizhevsky, Vinod Nair, y Geoffrey Hinton. Cifar-10 and cifar-100 (canadian institute for advanced research).
- [Knio9] Oliver Knill. *Probability Theory and Stochastic Processes with Applications*. Overseas Press, 2009.
- [Kol56] A.N. Kolmogorov. *Foundations Of The Theory Of Probability*. Chelsea Publishing Company, 1956.

- [KSH12] Alex Krizhevsky, Ilya Sutskever, y Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. En F. Pereira, C.J. Burges, L. Bottou, y K.Q. Weinberger, editores, *Advances in Neural Information Processing Systems*, volumen 25. Curran Associates, Inc., 2012.
- [LBBH98] Yann LeCun, L'eon Bottou, Yoshua Bengio, y Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [LBD⁺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.
- [LC05] Yann LeCun y Corinna Cortes. The mnist database of handwritten digits. 2005.
- [LJY24] Fei-Fei Li, Justin Johnson, y Serena Yeung. Cs231n: Deep learning for computer vision. Stanford University, Course Website, 2024. Lecture 4, Slide 45, <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. *Neurocomput.*, 470(C):443–456, January 2022.
- [LT24] Marc Lafon y Alexandre Thomas. Understanding the Double Descent Phenomenon in Deep Learning, March 2024. arXiv:2403.10459.
- [LZB21] Chaoyue Liu, Libin Zhu, y Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks, 2021.
- [Mal16] Stéphane Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203, April 2016.
- [MCK20] Yifei Min, Lin Chen, y Amin Karbasi. The curious case of adversarially robust models: More data can help, double descend, or hurt generalization, 2020.
- [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.
- [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.
- [Mur22] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.
- [Mur23] Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.
- [NGLK18] Cláudia Neves, Ignacio Gonzalez, John Leander, y Raid Karoumi. A new approach to damage detection in bridges using machine learning. páginas 73–84, 01 2018.
- [NKB⁺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.
- [NMB⁺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.
- [NZo8] M. A. Nilback y A. Zisserman. A visual vocabulary for flower classification. Available at <http://www.robots.ox.ac.uk/~vgg/data/flowers/102/>, 2008.
- [Opp95] Manfred Opper. Statistical mechanics of learning: Generalization. En *The Handbook of Brain Theory and Neural Networks*, páginas 922–925. Springer-Verlag, 1995.
- [Opp01] Manfred Opper. Learning to generalize. En *Frontiers of Life*, volumen 3, Part 2, páginas 763–775. Academic Press, 2001.

Bibliografía

- [PGM⁺19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, y Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. En *Advances in Neural Information Processing Systems* 32, páginas 8024–8035. Curran Associates, Inc., 2019.
- [Poo11] David Poole. *Álgebra lineal. Una introducción moderna*. Cengage Learning, 3 edición, 2011.
- [Pre94] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, european edición, 1994. Adapted by Darrel Ince.
- [Pri23] Simon J.D. Prince. *Understanding Deep Learning*. MIT Press, 2023.
- [RH21] Lars Ruthotto y Eldad Haber. *An introduction to deep generative modeling*, 2021.
- [Rip96] Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.
- [RSSW24] Hrithik Ravi, Clayton Scott, Daniel Soudry, y Yutong Wang. The implicit bias of gradient descent on separable multiclass data, 2024.
- [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⁺18] 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. En *International Conference on Learning Representations*, 2018.
- [Sch15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 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⁺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.
- [SGd⁺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):474001, October 2019.
- [SHN⁺24] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, y Nathan Srebro. The implicit bias of gradient descent on separable data, 2024.
- [SKR⁺23] Rylan Schaeffer, Mikail Khona, Zachary Robertson, Akhilan Boopathys, 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.
- [SLHS22] Sidak Pal Singh, Aurelien Lucchi, Thomas Hofmann, y Bernhard Schölkopf. Phenomenology of double descent in finite-width neural networks, 2022.
- [SLJ⁺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.
- [Str23] Gilbert Strang. *Introduction to Linear Algebra*. CUP, 6 edición, 2023.
- [Swa20] Swapna. Convolutional Neural Network | Deep Learning. <https://developersbreach.com/convolution-neural-network-deep-learning/>, 2020. [Recurso online, accedido el 19 de febrero de 2025].

Bibliografía

- [TGLM22] Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, y Gabriel F. Manso. The computational limits of deep learning, 2022.
- [Vap91] V. Vapnik. Principles of risk minimization for learning theory. En *Proceedings of the 5th International Conference on Neural Information Processing Systems*, NIPS'91, página 831–838, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.
- [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, jun 1989.
- [WH60] Bernard Widrow y Ted Hoff. Adaptive switching circuits. En *1960 IRE WESCON Convention Record, Part 4*, páginas 96–104. Institute of Radio Engineers, 1960.
- [YS24] Tian-Le Yang y Joe Suzuki. Dropout drops double descent, 2024.
- [ZBH⁺21] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, y Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM*, 64(3):107–115, February 2021.