UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

Estudio del uso de técnicas de Machine Learning y Deep Learning para la recomendación de circuitos eléctricos equivalentes para espectros de impedancias electroquímicas.

Trabajo Fin de Máster

Máster Universitario en Ingeniería Química

AUTOR/A: Sáez Pardo, Fermín

Tutor/a: Pérez Herranz, Valentín

Cotutor/a: Giner Sanz, Juan José

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE MÁSTER EN INGENIERÍA QUÍMICA

Estudio del uso de técnicas de Machine Learning y Deep Learning para la recomendación de circuitos eléctricos equivalentes para espectros de impedancias electroquímicas

AUTOR: Fermín Sáez Pardo

TUTOR: Valentín Pérez Herranz

COTUTOR: Juan José Giner Sanz

Curso académico: 2022/2023

Resumen

La espectroscopía de impedancias electroquímicas, EIS por sus siglas en inglés, es una técnica de caracterización electroquímica basada en la aplicación de un potencial (o corriente) senoidal y, la medida de la señal de corriente (o potencial) generada por el sistema. Esta técnica es ubicua en el campo de la electroquímica. Algunos campos de aplicación son: corrosión, pilas de combustible, baterías, sensores, etc. Este amplio abanico de aplicaciones se debe a su rapidez, sensibilidad y bajo coste, que la han convertido en una técnica omnipresente en la caracterización de sistemas electroquímicos.

El análisis de datos EIS se realiza principalmente mediante dos tipos de métodos. El primero consiste en el desarrollo de modelos físicos basados en las ecuaciones fundamentales que describen el sistema mientras que el segundo, consiste en el ajuste de los datos EIS a circuitos eléctricos equivalentes, que por su sencillez es la más utilizado.

La elección del circuito eléctrico equivalente es una etapa crítica en esta metodología y, se suele hacer en base a los patrones observados en los espectros EIS. En este contexto, Didby D. Macdonald propuso la creación del proyecto del genoma electroquímico, que tiene como objetivo la generación de una extensa base de datos, que guardaría información sobre la mayor cantidad y variedad de espectros electroquímicos posibles. Esta base de datos unida a una inteligencia artificial capaz de reconocer patrones y proponer el circuito eléctrico equivalente más plausible, supondría un gran avance en términos de tiempo y sistematicidad al identificar circuitos eléctricos equivalentes.

El objetivo de este Trabajo Final de Máster es aplicar diferentes algoritmos de Machine Learning y Deep Learning a la recomendación de circuitos eléctricos equivalentes para espectros EIS.

Para acometer el objetivo del Trabajo Final de Máster, se comenzó con una base de datos abierta que contiene espectros EIS etiquetados (i.e. espectros EIS asociados con el circuito eléctrico equivalente correspondiente). En una primera etapa, se realizó la depuración y ampliación de la base de datos mediante una revisión bibliográfica. En una segunda etapa, se utilizó la base de datos depurada y expandida para entrenar y optimizar diferentes algoritmos de Machine Learning y Deep Learning. Finalmente, se comparó el desempeño de los diferentes algoritmos entrenados y optimizados, en la tarea de recomendación de circuitos equivalentes a partir de espectros EIS.

Palabras Clave: Circuitos eléctricos equivalentes, Electroquímica, Deep Learning, Espectroscopía de Impedancias Electroquímicas, Inteligencia Artificial, Machine Learning.

Abstract

Electrochemical Impedance Spectroscopy (EIS) is an electrochemical technique based on the application of a sinusoidal potential (or current) and, the measurement of the current (or potential) generated by the system. This technique is applied in the whole field of electrochemistry. Examples of their applications are: corrosion, fuel cells, batteries, sensors, etc.

The analysis of EIS data is carried out by 2 methodologies. The first one consists in developing physical models based on the fundamental equations that describe the system. The second one consists in fitting EIS data to an equivalent circuit. The choice of the equivalent circuit is a critical stage in this methodology and, is generally carried out observing key patterns in EIS spectra.

The electrochemical genome project, proposed by Didby D. Macdonal, aims to generate a vast database containing all available electrochemical spectra. This database, working with an Artificial Intelligence (AI) able to recognize patterns and suggest equivalent circuits, would improve EIS spectrum analysis time and systematicity.

The goal of this Master's Thesis is to apply different Machine Learning and Deep Learning algorithms to build recommendation algorithms for suggesting equivalent circuits for EIS spectra.

In order to fullfill this goal: First, a database with labeled EIS spectrums (i.e. EIS spectrums associated with equivalent circuits) was obtained. Second, the database was depurated and expanded with bibliography data. Third, different algorithms were trained and, their hyperparameters were optimized. Finally, the performance of trained and optimized algorithms performance was compared.

Keywords: Artificial Intelligence, Deep Learning, Electrochemical impedance Spectroscopy, Equivalent circuits, Electrochemistry, Machine Learning.

Resum

La espectroscopía d'impedàncies electroquímiques, EIS per les seues sigles en anglés, és una tècnica de caracterització electroquímica basada en l'aplicació d'un potencial (o corrent) senoidal i, la mesura del senyal de corrent (o potencial) generada pel sistema. Aquesta tècnica és ubiqua en el camp de l'electroquímica. Alguns camps d'aplicació són: corrosió, piles de combustible, bateries, sensors, etc. Aquest ampli ventall d'aplicacions es deu a la seua rapidesa, sensibilitat i baix cost, que l'han convertida en una tècnica omnipresent en la caracterització de sistemes electroquímics.

L'anàlisi de dades EIS es realitza principalment mitjançant dos tipus de mètodes. El primer, consisteix en el desenvolupament de models físics basats en les equacions fonamentals que descriuen el sistema. El segon, consisteix en l'ajust dades EIS a circuits elèctrics equivalents, que per senzillesa és la més utilitzada.

L'elecció del circuit elèctric equivalent és una etapa crítica en aquesta metodologia i, se sol fer sobre la base dels patrons observats en els espectres EIS. En aquest context, Didby D. Macdonald va proposar la creació del projecte del genoma electroquímic que, té com a objectiu la generació d'una extensa base de dades, que guardaria informació sobre la major quantitat i varietat d'espectres electroquímics possibles. Aquesta base de dades unida a una intel·ligència artificial capaç de reconéixer patrons i proposar el circuit elèctric equivalent més plausible, suposaria un gran avanç en termes de temps i sistematicitat en identificar circuits elèctrics equivalents.

L'objectiu d'aquest Treball Final de Màster és aplicar diferents algorismes de Machine Learning i Deep Learning a la recomanació de circuits elèctrics equivalents per a espectres EIS.

Per a escometre l'objectiu del Treball Final de Màster, es va començar amb una base de dades oberta que conté espectres EIS etiquetatges (i.e. espectres EIS associats amb el circuit elèctric equivalent corresponent). En una primera etapa, es va realitzar la depuració i ampliació de la base de dades mitjançant una revisió bibliogràfica. En una segona etapa, es va utilitzar la base de dades depurada i expandida per a entrenar i optimitzar diferents algorismes de Machine Learning i Deep Learning. Finalment, es va comparar l'acompliment dels diferents algorismes entrenats i optimitzats, en la tasca de recomanació de circuits equivalents a partir d'espectres EIS.

Paraules Clau: Circuits elèctrics equivalents, Electroquímica, Deep Learning, Espectroscopía d'Impedàncies Electroquímiques, Intelligència artificial, Machine Learning.

Antecedentes

Justificación

El presente Trabajo Final de Máster (TFM) tiene el objetivo docente de validar al alumno en los conocimientos necesarios para obtener el Máster Universitario en Ingeniería Química (MUIQ) de la Universidad Politécnica de Valencia (UPV).

En consecuencia, la temática de este TFM está relacionada con 2 áreas de conocimiento impartidas en el MUIQ. Por un lado, la electroquímica, que está presente en las asignaturas de Diseño Avanzado de Reactores II, Ingeniería Electroquímica, y Corrosión. Por otro lado, el desarrollo de Inteligencias Artificiales (IA) en el ámbito de la Ingeniería Química, que está presente en las asignaturas de Modelización, Simulación y Optimización de Procesos Químicos, y Métodos Avanzados de diseño, Simulación y Análisis de Procesos Químicos.

Motivación

La elección de este TFM por parte del alumno ha estado motivada por la curiosidad y ganas de aprender sobre Electroquímica e IA. Estos campos, a juicio del alumno, son (en el caso de la electroquímica) o serán (en el caso de la IA) campos imprescindibles en el currículum de un Ingeniero Químico.

Relación Objetivos de Desarrollo Sostenible

Este TFM se centra en la creación de una herramienta para la recomendación de circuitos eléctricos equivalentes para espectros de impedancias electroquímicas procedentes de la técnica de Espectroscopía de Impedancias Electroquímicas. Esta técnica es omnipresente en el campo de la electroquímica y, por lo tanto, puede ser relacionada con diversos Objetivos de Desarrollo Sostenible (ODS) a través de las aplicaciones de la electroquímica:

- ODS 2, Hambre cero: Producción de nitrógeno amoniacal para la generación de fertilizantes in situ mediante técnicas electroquímicas, lo que está relacionado con las metas 2.3 (duplicar

la productividad agrícola e ingresos de los productores de alimentos a pequeña escala), y 2.4 (aplicar prácticas agrícolas que aumenten la productividad y la producción).

- ODS 6, Agua limpia y saneamiento: La aplicación de técnicas electroquímicas para la oxidación de compuestos se conoce como electroxidación. Esta técnica permite la degradación de contaminantes hídricos, lo que está relacionado con las metas 6.1 (acceso universal al agua potable), 6.3 (mejorar la calidad del agua reduciendo la contaminación), y 6.6 (proteger y restablecer ecosistemas relacionados con el agua).
- ODS 7, Energía asequible y no contaminante: La electroquímica es la base para el desarrollo de novedosas formas de almacenamiento y generación de energía, lo que está relacionado con las metas 7.1 (acceso universal a servicios energéticos asequibles, fiables y modernos), y 7.2 (aumentar la proporción de energía renovable en el conjunto de fuentes energéticas).
- ODS 12, Producción y consumo responsables: La Electroquímica es la base para la recuperación metales presentes en efluentes acuosos de diferentes industrias y procesos (e.g. industria minera, procesos de recubrimiento de metales, etc), lo que está relacionado con las metas 12.2 (gestión sostenible y uso eficiente de los recursos naturales) y 12.5 (reducción de la generación de deshechos mediante actividades de reducción, reciclado y reutilización).

Los niveles de relación (i.e. Fuerte, Moderado, o Escaso) para cada meta relacionada con este TFM se muestran en la tabla 1.

Tabla 1: Niveles de relación para cada meta relacionada con este TFM.

ODS	Meta	Nivel de relación
2	2.3	Escaso
	2.4	Escaso
6	6.1	Moderado
	6.3	Moderado
	6.6	Moderado
7	7.1	Fuerte
	7.2	Fuerte
12	12.2	Moderado
	12.5	Moderado

Nomenclatura

Letras latinas

A	Área	m^2
b	Sesgo de la red neuronal	
C	Capacitancia	F
C'	Box Constrain	
C_{Ox}	Concentración de especie oxidada	$\mathrm{mol} \cdot \mathrm{m}^{-3}$
C_{Red}	Concentración de especie reducida	$\mathrm{mol} \cdot \mathrm{m}^{-3}$
D_{Red}	Difusividad de la especie reducida	$\mathrm{m}^2 \cdot \mathrm{s}^{-1}$
E	Diferencia de voltaje	V
EE	Función de coste de entrecruzamiento entrópico	
F	Constante de Faraday	$\mathrm{C} \cdot \mathrm{mol}_{\mathrm{e}^-}^{-1}$
f	Frecuencia	Hz
f_{ac}	Función de activación	
f_{cost}	Función de coste	
I	Corriente	A
i	Densidad de corriente	$\mathrm{A} \cdot \mathrm{m}^{-2}$
i_0	Densidad de corriente de intercambio	$\mathrm{A} \cdot \mathrm{m}^{-2}$
IG	Función de coste Índice de Gini	
ILR	Initial Learn Rate	
j	Unidad imaginaria (i.e. $\sqrt{-1}$)	
L	Inductancia	H

n	Número de electrones intercambiados	$\mathrm{mol_{e^-}} \cdot \mathrm{mol}^{-1}$
$\mathbb{P}$	Probabilidad	
Q	Capacitancia efectiva del CPE	$\mathrm{S}^{\alpha^*} \cdot \Omega^{-1} \cdot \mathrm{m}^{-2}$
$\Re$	Parte real	
R	Resistencia	Ω
R_G	Constante de los gases ideales	$\mathrm{J} \cdot \mathrm{mol}^{-1} \cdot \mathrm{K}^{-1}$
s	Información agregada	
T	Temperatura	K
t	Tiempo	s
$\mathbf{W}$	Matriz de pesos de la red neuronal	
$\vec{w}$	Vector normal	
$\mathbf{X}$	Matriz de ejemplos	
$\hat{y}$	Predicción realizada por un clasificador	
$\vec{Y}$	Vector de etiquetas	
y	Distancia al electrodo	m
y_{input}	Información de entrada a una neurona	
y_{output}	Información de salida de una neurona	
$\tilde{Z}$	Impedancia	$\Omega \cdot \mathrm{m}^2$
$\tilde{z}$	Impedancia local	Ω
$\tilde{Z}'$	Parte real de la impedancia	$\Omega \cdot \mathrm{m}^2$
$\tilde{Z}''$	Parte imaginaria de la impedancia	$\Omega \cdot \mathrm{m}^2$
$\tilde{Z}_{CPE}$	Impedancia de elemento de fase constante (CPE)	$\Omega \cdot \mathrm{m}^2$
$\tilde{Z}_D$	Impedancia de transferencia de materia	$\Omega \cdot \mathrm{m}^2$
$\tilde{Z}_f$	Impedancia farádica	$\Omega \cdot \mathrm{m}^2$
$\tilde{Z}_t$	Impedancia	Ω
$\tilde{Z}_W$	Impedancia de Warburg	$\Omega \cdot \mathrm{m}^2$

Letras griegas

α	Coeficiente de transferencia de carga
α^*	Coeficiente de ajuste del CPE
γ	Momento

ψ	Vector de parámetros optimizables	
θ	Concentración adimensional	
ω	Frecuencia angular	$\mathrm{rad} \cdot \mathrm{s}^{-1}$
ξ	Parámetro de regularización del algoritmo support vector machine	

Índice general

Índice de figuras

Índice de tablas

Parte I

Memoria

Capítulo 1

Introducción

1.1 Historia de la Espectroscopía de Impedancias Electroquímicas

La historia de esta técnica se remonta a 1872, cuando Heaviside aplicó las transformadas de Laplace a los transitorios de circuitos eléctricos, asentando las bases del estudio de circuitos eléctricos y acuñando los términos de inductancia, capacitancia e impedancia (Macdonald 2006). Sin embargo, su aplicación a sistemas electroquímicos no se produjo hasta 1894 (Orazem y Tribollet 2008), cuando Nernst la empleó para la medida de constantes dieléctricas de electrolitos.

En los años posteriores, otros autores adoptaron la metodología de Nernst (Nernst 1894) para la medida de las propiedades dieléctricas de otros sistemas, y la medida de resistencia de celdas galvánicas. No obstante, el primer gran resultado teórico se atribuye a la teoría de Warburg publicada en 1899. En las décadas siguientes, se avanzó y se expandió el campo de aplicación a la caracterización de electrodos y sistemas biológicos (Fricke y Morse 1925).

En la década de 1930, Fricke (Fricke 1932) constató la relación de la impedancia con la frecuencia elevada a un parámetro experimental para algunos sistemas, lo que condujo al desarrollo del elemento de fase constante o, CPE por sus siglas en inglés (Constant Phase Element), de la mano de Cole y Cole (K. S. Cole y R. H. Cole 1941). A su vez, Frumkim (Frumkin 1940) estudió la estructura de la doble capa y, Randles (Randles 1947) propuso un circuito eléctrico equivalente, que modelaba la cinética de una reacción electroquímica elemental.

Con unas bases sólidas, la técnica se aplicó a sistemas más complejos en los años siguientes. Sin embargo, la técnica no alcanzó su estatus actual hasta la aparición de los analizadores de respuesta de frecuencia, o FRA por sus siglas en inglés (Frecuency Response Analyzer), que permitieron la medida de impedancias en un rango más amplio de frecuencias.

Contemporáneamente a estos avances en la instrumentación de la técnica, se desarrollaron diferentes herramientas matemáticas, como las relaciones de Kramers-Kronig, que relacionan la parte real e imaginaria de la impedancia a través de las restricciones de los sistemas lineales; y

la regresión compleja no lineal, que permite el ajuste de espectros de impedancias a circuitos eléctricos equivalentes.

1.2 Aplicaciones de la Espectroscopía de Impedancias Electroquímicas

Las aplicaciones de la Espectroscopía de Impedancias Electroquímicas son muy amplias: se pueden encontrar ejemplos de su aplicación en caracterización de baterías de ion-litio (Meddings et al. 2020), pilas de combustible (Brunetto, Moschetto y Tina 2009), procesos de corrosión (Jüttner 1990), sensores (Poghossian y Schöning 2020), baterías de flujo redox (Derr et al. 2016) y sistemas bioquímicos (Vidaković-Koch et al. 2013), entre otros muchos campos de aplicación.

La razón de su ubicuidad son las numerosas ventajas que ofrece entre las que se encuentra su rapidez, sensibilidad y bajo coste. Esto ha llevado a un crecimiento exponencial del número de publicaciones que emplean esta técnica, como se muestra en la figura 1.1.

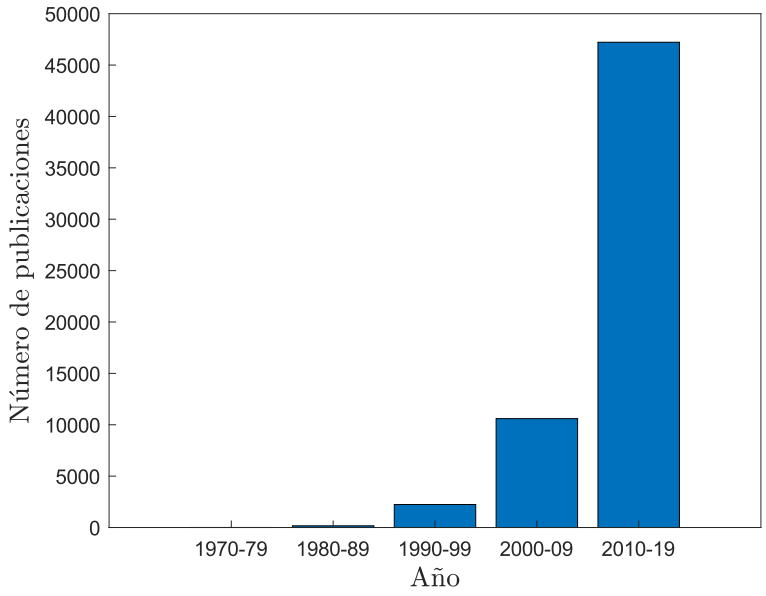


Figura 1.1: Evolución temporal del número de publicaciones relacionadas con la Espectroscopía de Impedancias Electroquímicas. Datos obtenidos mediante la búsqueda de "electrochemical impedance" como palabras clave el 10 de noviembre de 2022 en el buscador SCOPUS.

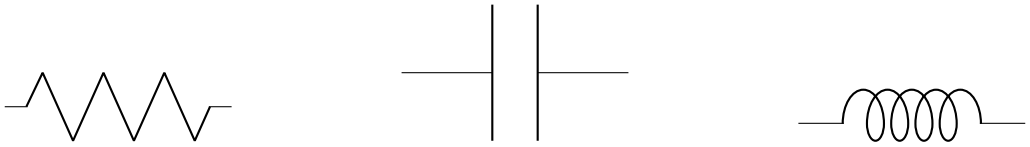
1.3 Impedancias eléctricas

En los circuitos eléctricos, existen principalmente 3 elementos diferentes: elementos resistivos, capacitivos e inductivos, cuya simbología se muestra en la figura 1.2. Los elementos resistivos se caracterizan por no producir un desfase entre el voltaje (E) y la intensidad (I), simplemente un cambio de amplitud dado por la ley de Ohm:

$$R = \frac{E}{I} \tag{1.1}$$

Por otro lado, los elementos capacitivos e inductivos son capaces de almacenar energía en forma de carga o campo magnético, lo que se conoce como capacitancia (C) e inductancia (L), respectivamente. Dichas magnitudes están presentes en las ecuaciones diferenciales que describen el comportamiento de dichos elementos:

$$I = C \cdot \frac{dE}{dt} \tag{1.2}$$



(a) Elemento resistivo **(b)** Elemento capacitivo **(c)** Elemento inductivo

Figura 1.2: Elementos eléctricos más comunes

$$E = L \cdot \frac{dI}{dt} \tag{1.3}$$

Como se deduce de estas ecuaciones diferenciales ordinarias (EDOs), en régimen estacionario, el elemento capacitivo se comporta como un circuito abierto; mientras que el elemento inductivo lo hace como un cortocircuito.

Ambas EDOs son lineales y, por tanto, la aplicación de una diferencia de voltaje (E) sinuosidal genera una intensidad (I) sinusoidal de la misma frecuencia angular (ω), pero con un desfase (ϕ) y distinta amplitud. Además, en este tipo de casos la introducción de números complejos simplifica notablemente la resolución de las EDOs.

Por ejemplo, la aplicación de una diferencia de voltaje sinuosidal con una amplitud (E_{max}) y frecuencia angular (ω) se puede expresar como:

$$E = E_{max} \cdot cos(\omega \cdot t) = E_{max} \cdot \Re\{e^{j \cdot \omega \cdot t}\} \tag{1.4}$$

Donde $j = \sqrt{-1}$ es la unidad imaginaría y $\Re\{\}$ es el operador parte real.

Si dicho voltaje sinuosidal se aplica a un elemento capacitivo, se tendría:

$$I = C \cdot \frac{d(E_{max} \cdot \Re\{e^{j \cdot \omega \cdot t}\})}{dt} \tag{1.5}$$

Derivando la diferencia de voltaje con el tiempo se obtiene:

$$I = \Re\left\{C \cdot E_{max} \cdot \omega \cdot j \cdot e^{j \cdot \omega \cdot t}\right\} = \Re\left\{C \cdot E_{max} \cdot \omega \cdot e^{j(\omega \cdot t + \frac{\pi}{2})}\right\} = C \cdot E_{max} \cdot \omega \cdot cos\left[\omega \cdot t + \frac{\pi}{2}\right] \tag{1.6}$$

Por lo tanto, un condensador provoca un desfase entre la intensidad y voltaje de 90°. Además, la división de diferencia de voltaje entre la intensidad da:

$$\frac{E}{I} = \frac{E_{max} \cdot \Re\{e^{j \cdot \omega \cdot t}\}}{\Re\{C \cdot E_{max} \cdot \omega \cdot e^{j(\omega \cdot t + \frac{\pi}{2})}\}} = \frac{E}{I} = \frac{E_{max} \cdot cos(\omega \cdot t)}{C \cdot E_{max} \cdot \omega \cdot cos(\omega \cdot t + \frac{\pi}{2})} = \frac{cos(\omega \cdot t)}{-C \cdot \omega \cdot sen(\omega \cdot t)} \tag{1.7}$$

Aunque, la expresión 1.7 parezca la resistencia de un elemento capacitivo no lo es, ya que, la resistencia de un elemento debe ser una constante en el dominio del tiempo y, la expresión 1.7 no cumple esta condición.

Un fasor es la representación de un número complejo en forma polar, es decir, la representación mediante su módulo y ángulo que forma con el eje, como se muestra en la figura 1.3. La relación de los fasores de diferencia de voltaje ($\tilde{V}$) e intensidad ($\tilde{I}$) sí que origina una constante en el domino del tiempo.

Por lo tanto, los fasores de diferencia de voltaje e intensidad se definen como:

$$\tilde{E} \equiv E_{max} \cdot e^{0 \cdot j} \tag{1.8}$$

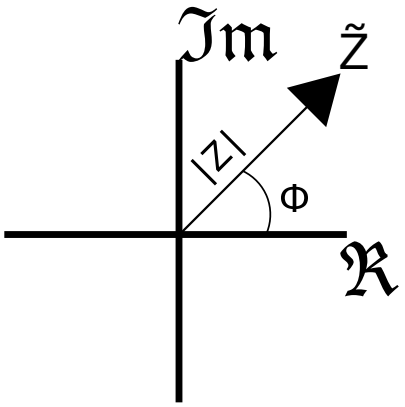
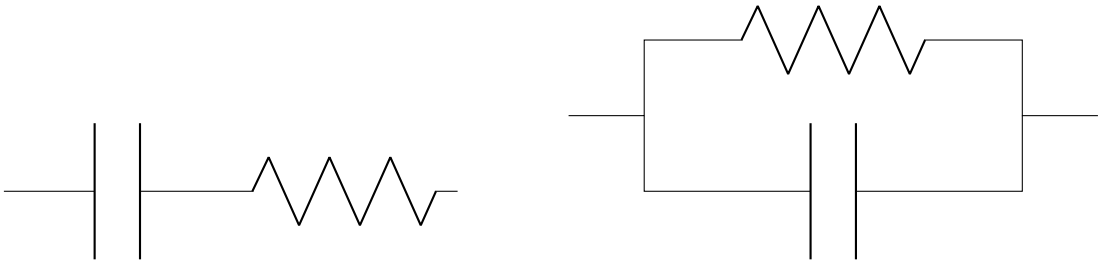


Figura 1.3: Representación fasorial



(a) Serie **(b)** Paralelo

Figura 1.4: Conexiones de componentes eléctricos

$$\tilde{I} \equiv E_{max} \cdot \omega \cdot C \cdot e^{\frac{\pi}{2} \cdot j} \tag{1.9}$$

Y su división es el fasor impedancia total ($\tilde{Z}_t$) de un elemento capacitivo:

$$\tilde{Z}_t = \frac{E_{max} \cdot e^0}{E_{max} \cdot \omega \cdot C \cdot e^{\frac{\pi}{2} \cdot j}} = \frac{1}{\omega \cdot C \cdot j} = -\frac{j}{C \cdot \omega} \tag{1.10}$$

Operando de manera análoga al condensador se puede llegar a la expresión del fasor impedancia para un elemento inductivo:

$$\tilde{Z}_t = j \cdot \omega \cdot L \tag{1.11}$$

El término real del fasor impedancia hace referencia a la resistencia y, el término imaginario hace referencia a la reactancia. La impedancia describe cualquier elemento que posea una combinación de resistencia y reactancia y, por lo tanto, el concepto de impedancia es una generalización de la ley de Ohm.

En cuanto a la combinación de diferentes elementos (resistivos, capacitivos e inductivos) existen 2 maneras principales de conexión: serie y paralelo. La conexión en serie de dos elementos se da cuando por ambos elementos pasa la misma intensidad. Por otro lado, la conexión en paralelo surge cuando la diferencia de voltaje entre los bornes de los elementos es la misma, como se muestra en la figura 1.4.

Por un lado, las ecuaciones que surgen de la disposición en serie de dos elementos (siendo $\tilde{E}_1$, $\tilde{I}_1$ $\tilde{E}_2$, $\tilde{I}_2$, $\tilde{E}_T$ y $\tilde{I}_T$ la caída de voltaje e intensidad del elemento 1, 2 y total, respectivamente) vienen dadas por:

$$\tilde{E}_T = \tilde{E}_1 + \tilde{E}_2 \tag{1.12}$$

$$\tilde{I}_T = \tilde{I}_1 = \tilde{I}_2 \tag{1.13}$$

Por lo tanto la impedancia total ($\tilde{Z}_{tT}$) es:

$$\tilde{Z}_{tT} = \frac{\tilde{E}_1}{\tilde{I}_1} + \frac{\tilde{E}_2}{\tilde{I}_2} = \tilde{Z}_{t1} + \tilde{Z}_{t2} \tag{1.14}$$

Por otro lado, la disposición en paralelo impone las siguientes restricciones:

$$\tilde{E}_T = \tilde{E}_1 = \tilde{E}_2 \tag{1.15}$$

$$\tilde{I}_T = \tilde{I}_1 + \tilde{I}_2 \tag{1.16}$$

Por lo tanto, la impedancia total:

$$\tilde{Z}_{tT} = \frac{\tilde{E}_T}{\tilde{I}_1 + \tilde{I}_2} = \frac{\tilde{E}_T}{\frac{\tilde{E}_T}{\tilde{Z}_{t1}} + \frac{\tilde{E}_T}{\tilde{Z}_{t2}}} = \frac{1}{\frac{1}{\tilde{Z}_{t1}} + \frac{1}{\tilde{Z}_{t2}}} \tag{1.17}$$

1.4 Fundamentos de Espectroscopía de Impedancias Electroquímicas

En la sección 1.3 se ha mostrado que el término impedancia es una generalización del término resistencia. Sin embargo, es posible generalizar todavía más e interpretar la impedancia como un caso particular de funciones de transferencia, como se ilustra en la figura 1.5.

Las funciones de transferencia son una manera compacta de representar la relación dinámica de entrada/salida de un sistema lineal (Åström y Murray 2021). El uso de funciones de transferencia impone las condiciones de la teoría de sistemas lineales (Macdonald 2006):

- El sistema debe ser lineal.
- El sistema debe ser estable.
- El sistema debe ser causal.
- La impedancia debe ser finita.

Por ejemplo, las impedancias eléctricas mostradas en la sección 1.3 están descritas por ecuaciones diferenciales lineales (condición de linealidad), el sistema alcanza un estado estacionario tras la

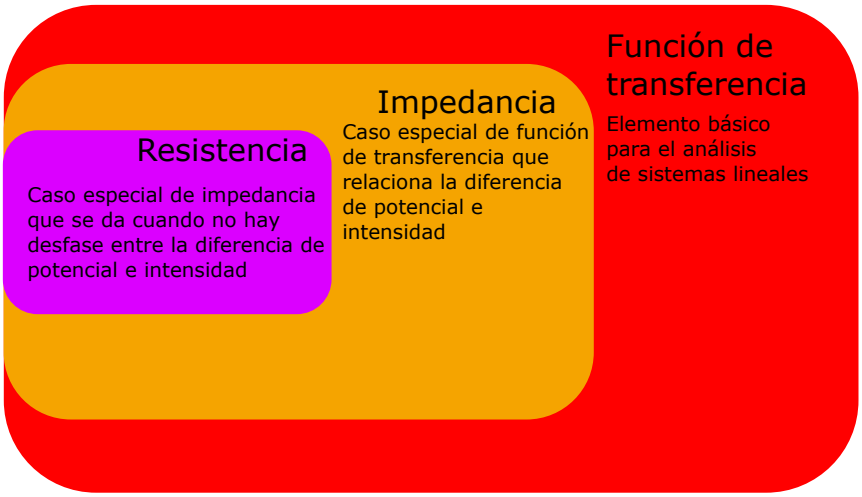


Figura 1.5: Relación conceptual de resistencia, impedancia y función de transferencia

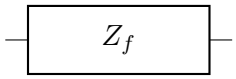
aplicación de una perturbación (condición de estabilidad), la perturbación no genera una respuesta en el sistema antes de ser aplicada (condición de casualidad) y las impedancias resultantes no tienen un valor infinito (condición de finitud).

Con esto en mente, el concepto de impedancia puede ser expandido a cualquier contexto en el que entren en juego las magnitudes voltaje e intensidad, y se cumplan las condiciones anteriormente mencionadas. Uno de estos contextos son los sistemas electroquímicos, siendo esta la base teórica de la Espectroscopía de Impedancias Electroquímicas.

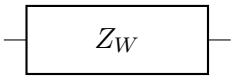
Por lo tanto, la Espectroscopía de Impedancias Electroquímicas se basa en la excitación de sistemas electroquímicos mediante señales de voltaje o intensidad sinusoidales de diferentes frecuencias y, la medida de intensidad o diferencia de voltajes resultantes. Dicha medida proporciona una triada de valores (f, Z' y Z''), que corresponde con el espectro de impedancias electroquímicas.

1.4.1 Elementos característicos en Espectroscopía de Impedancias Electroquímicas

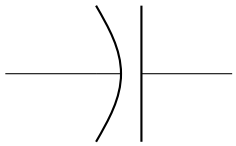
Al igual que en el contexto de circuitos eléctricos existen elementos característicos (resistencias, condensadores, bobinas), en el ámbito electroquímico se tienen las resistencias farádicas (asociadas a la resistencia de transferencia de carga de los electrones en el electrodo) (Orazem y Tribollet 2008), resistencia de Warburg (asociada a limitaciones de transferencia de materia) (Orazem y Tribollet 2008), capacitancias de doble capa (asociado a la polarización del electrolito en las inmediaciones del electrodo) (Newman y Balsara 2021), y elementos de fase constante o CPE por sus siglas en inglés (Constant Phase Element) (asociado a las heterogeneidades del electrodo) (Orazem y Tribollet 2008), siendo sus símbolos mostrados en la figura 1.6.



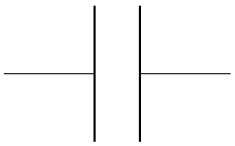
(a) Resistencia farádica



(b) Resistencia de Warburg



(c) Elemento de fase constante (CPE)



(d) Capacitancia de doble capa

Figura 1.6: Símbolos para los elementos característicos más comunes en Espectroscopía de Impedancias Electroquímicas

Impedancia farádica

Estos elementos aparecen asociados a reacciones farádicas (i.e. de intercambio de electrones). Por ejemplo, el estudio de la reacción electroquímica mostrada en R1 da lugar a la resistencia farádica, que está asociada a la transferencia de electrones.

$$\mathrm{Red} \rightleftharpoons \mathrm{Ox} + \mathrm{n_{e^-}} \tag{R1}$$

La cinética de esta reacción electroquímica viene dada por la ecuación de Butler-Volmer:

$$i = i_0 \cdot \left(\frac{C_{Red}|_e}{C_{Red}|_\infty} \cdot e^{\frac{\alpha \cdot n \cdot F}{R_G \cdot T} \cdot (E - E_{eq})} - \frac{C_{Ox}|_e}{C_{Ox}|_\infty} \cdot e^{\frac{-(1-\alpha) \cdot n \cdot F}{R_G \cdot T} \cdot (E - E_{eq})} \right) \tag{1.18}$$

Donde cada sumando de la ecuación 1.18 hace referencia a uno de los sentidos de la reacción electroquímica R1 (según la convención adoptada, el término positivo a la reacción de oxidación y el negativo a la de reducción).

La terminología utilizada es la siguiente: α es el coeficiente de transferencia de carga, n es el número de electrones intercambiados, F es la constante de Faraday, i_0 es la densidad de corriente de intercambio, R_G es la constante de los gases ideales, T temperatura, $C_{Red}|_e$ y $C_{Ox}|_e$ son las concentraciones en la superficie de la especie reducida y oxidada respectivamente, y $C_{Red}|_\infty$ y $C_{Ox}|_\infty$ son las concentraciones en el seno de la disolución de las especies reducida y oxidada respectivamente.

Considerando voltajes lo suficientemente elevados para despreciar la reacción de reducción:

$$i \simeq i_0 \cdot \frac{C_{Red}|_e}{C_{Red}|_\infty} e^{\frac{\alpha \cdot n \cdot F}{R_G \cdot T} \cdot (E - E_{eq})} \tag{1.19}$$

En ausencia de limitaciones de transferencia de transferencia de materia, la razón entre la concentración de la especie reducida en el electrodo y en el seno de la disolución puede ser considerada constante, lo que permite englobarla dentro de la constante k^*:

$$i = \overbrace{i_0 \cdot \frac{C_{Red}|_e}{C_{Red}|_\infty} \cdot e^{\frac{-\alpha \cdot n \cdot F}{R_G \cdot T} \cdot E_{eq}}}^{k^*} \cdot e^{\frac{\alpha \cdot n \cdot F}{R_G \cdot T} \cdot E} \tag{1.20}$$

Además, las constantes del término exponencial también pueden ser agrupadas bajo la constante b, como se muestra en la ecuación 1.21.

$$i = k^* \cdot e^{\overbrace{\frac{\alpha \cdot n \cdot F}{R_G \cdot T}}^{b} \cdot E} \tag{1.21}$$

La ecuación 1.21 muestra la dependencia exponencial de la densidad de corriente (i) con la diferencia de voltaje (E) para una reacción farádica. Esto viola la condición de linealidad de las impedancias y, por lo tanto, es necesario linealizar la ecuación 1.21 mediante el desarrollo en serie de Taylor.

Para ello, se expresan las funciones sinuosidales de densidad de corriente y diferencia de voltaje en función de su valor medio ($\bar{E}$ y $\bar{i}$), y su fasor ($\tilde{E}$ y $\tilde{i}$):

$$i = \bar{i} + \Re\{\tilde{i} \cdot e^{j\omega \cdot t}\} \tag{1.22}$$

$$E = \bar{E} + \Re\{\tilde{E} \cdot e^{j\omega \cdot t}\} \tag{1.23}$$

Aplicando el desarrollo en serie de Taylor a la ecuación 1.21 alrededor del valor medio para la intensidad, se obtiene la ecuación 1.24.

$$i = \bar{i} + \left.\frac{di}{dE}\right|_{\bar{E}} \cdot (E - \bar{E}) \tag{1.24}$$

Sustituyendo las expresiones 1.22 y 1.23, en 1.24 y derivando:

$$\bar{i} + \Re\{\tilde{i} \cdot e^{j \cdot \omega \cdot t}\} = \bar{i} + (k^* \cdot b \cdot e^{b \cdot \bar{E}}) \cdot (\Re\{\tilde{E} \cdot e^{j \cdot \omega \cdot t}\}) \tag{1.25}$$

Identificando términos, se obtiene:

$$\Re\{\tilde{i}\} = \Re\{k^* \cdot b \cdot e^{b \cdot \bar{E}} \cdot \tilde{E}\} \tag{1.26}$$

Finalmente, aplicando la ley de Ohm generalizada se obtiene:

$$\tilde{Z}_f = \frac{\tilde{E}}{\tilde{i}} = \frac{1}{k^* \cdot b \cdot e^{b \cdot \bar{E}}} \tag{1.27}$$

Atendiendo a la ecuación 1.27 se puede concluir que, la resistencia farádica se comporta como un elemento resistivo puro, ya que, el fasor $\tilde{Z}_f$ solo tiene componente real.

Impedancia de Warburg

La impedancia farádica ($\tilde{Z}_f$) aparece siempre que se dan reacciones electroquímicas. Sin embargo, las reacciones electroquímicas pueden tener asociadas una resistencia adicional, originada por limitaciones de transferencia de materia, que es conocida como impedancia de Warburg.

Para el cálculo de esta resistencia se parte de la ecuación 1.21 y se extrae la concentración en las inmediaciones del electrodo de la constante, ya que, la concentración en este caso no es constante:

$$i = k \cdot C_{Red}|_e \cdot e^{b \cdot E} \tag{1.28}$$

Aplicando el desarrollo en serie de Taylor multivariable, se obtiene:

$$i = \bar{i} + \left.\frac{di}{dE}\right|_{\bar{E}} \cdot (E - \bar{E}) + \left.\frac{di}{dC_{Red}|_e}\right|_{\bar{C}_{Red}|_e} \cdot (C_{Red}|_e - \bar{C}_{Red}|_e) \tag{1.29}$$

Realizando las derivadas, se obtiene:

$$i = \bar{i} + k \cdot \bar{C}_{Red}|_e \cdot b \cdot e^{b \cdot \bar{E}} \cdot \Re\{\tilde{E} \cdot e^{j \cdot \omega t}\} + k \cdot e^{b \cdot \bar{E}} \cdot \Re\{\tilde{C}_{Red}|_e \cdot e^{j \cdot \omega \cdot t}\} \tag{1.30}$$

Para este caso, es necesario relacionar el fasor de concentración de la especie reducida en la superficie del electrodo ($\tilde{C}_{Red}|_e$) con la intensidad, mediante la ley de Fick:

$$i = n \cdot F \cdot D_{Red} \cdot \left.\frac{dC_{Red}|_e}{dy}\right|_{y=0} \tag{1.31}$$

Sustituyendo se obtiene:

$$\bar{i} + \Re\{\tilde{i} \cdot e^{j \cdot \omega \cdot t}\} = n \cdot F \cdot D_{Red} \cdot \left(\left.\frac{d\bar{C}_{Red}|_e}{dy}\right|_{y=0} + \left.\frac{d\Re\{\tilde{C}_{Red}|_e \cdot e^{j \cdot \omega \cdot t}\}}{dy}\right|_{y=0} \right) \tag{1.32}$$

Por identificación en la ecuación 1.32:

$$\bar{i} = n \cdot F \cdot D_{Red} \cdot \left.\frac{d\bar{C}_{Red}|_e}{dy}\right|_{y=0} \tag{1.33}$$

$$\Re\{\tilde{i} \cdot e^{j \cdot \omega \cdot t}\} = n \cdot F \cdot D_{Red} \cdot \left.\frac{d\Re\{\tilde{C}_{Red}|_e \cdot e^{j \cdot \omega \cdot t}\}}{dy}\right|_{y=0} \tag{1.34}$$

Realizando el cambio de variable:

$$\tilde{\theta} = \frac{\tilde{C}_{Red}}{\tilde{C}_{Red}|_e} \tag{1.35}$$

Se obtiene:

$$\Re\{\tilde{i}\} = \Re\{n \cdot F \cdot D_{Red} \cdot \tilde{C}_{Red}|_e \cdot \tilde{\theta}'(0)\} \tag{1.36}$$

Despejando $\tilde{C}_{Red}|_e$, se obtiene:

$$\Re\{\tilde{C}_{Red}|_e\} = \Re\left\{\frac{i}{n \cdot F \cdot D_{Red}\tilde{\theta}'(0)}\right\} \tag{1.37}$$

Sustituyendo la ecuación 1.37 en 1.30, se obtiene:

$$\Re\{i\} = \Re\left\{k \cdot \bar{C}_{Red}|_e \cdot b \cdot e^{b\cdot\bar{E}} + k \cdot e^{b\cdot\bar{E}} \cdot \frac{\tilde{i}}{n \cdot F \cdot D_{Red} \cdot \tilde{\theta}'(0)}\right\} \tag{1.38}$$

Finalmente, se obtiene que la impedancia de una sistema electroquimico con reacción farádica y limitaciones de transferencia de materia es:

$$\tilde{Z}\frac{\tilde{E}}{\tilde{i}} = \overbrace{\frac{1}{k \cdot \bar{C}_{Red}|_e \cdot b \cdot e^{b\cdot\bar{E}}}}^{\tilde{Z}_f} + \overbrace{\frac{1}{n \cdot F \cdot D_{Red} \cdot \bar{C}_{Red}|_e \cdot b} \cdot \frac{-1}{\tilde{\theta}'(0)}}^{\tilde{Z}_D} \tag{1.39}$$

Como se puede observar, la impedancia global esta formada por la resistencia farádica ($\tilde{Z}_f$) y la resistencia asociada a la transferencia de materia ($\tilde{Z}_D$), ambas en serie.

Sin embargo, $\tilde{Z}_D$ es simplemente una expresión general para la obtención de la impedancia de transferencia de materia. Para obtener la impedancia de Warburg es necesario explicitar el término $\tilde{\theta}'(0)$.

En función del caso de estudio existen diferentes formas de explicitar $\tilde{\theta}'(0)$, eligiendo el caso de una capa límite infinita (caso más sencillo) las condiciones de contorno son:

$$y = 0 \rightarrow \theta = 0 \tag{1.40}$$

$$y = \infty \rightarrow \theta = 1 \tag{1.41}$$

Aplicando la segunda ley de Fick, se obtiene:

$$\frac{\partial\theta}{\partial t} - D_{Red} \cdot \frac{\partial^2\theta}{\partial y^2} = 0 \tag{1.42}$$

La ecuación 1.42 no tiene un estado estacionario, lo que viola la condición de estabilidad. Sin embargo, es posible suponer que tras un tiempo, el perfil de concentraciones en las inmediaciones alcance un estado pseudoestacionario, que permita la medición de la impedancia.

Con esto en mente, al sustituir en la ecuación 1.42, se obtiene 1.43.

$$\frac{\partial\Re\{\tilde{\theta} \cdot e^{j\cdot\omega\cdot t}\}}{\partial t} - D_{Red} \cdot \frac{\partial^2\Re\{\tilde{\theta} \cdot e^{j\cdot\omega\cdot t}\}}{\partial y^2} = 0 \tag{1.43}$$

Operando, se obtiene:

$$j \cdot \omega \cdot \tilde{\theta} - D_{Red} \cdot \frac{\partial^2\tilde{\theta}}{\partial y^2} = 0 \tag{1.44}$$

Reordenando términos, se obtiene la ecuación diferencial de segundo orden siguiente:

$$\frac{\partial^2 \tilde{\theta}}{\partial y^2} = \frac{j \cdot \omega \cdot \tilde{\theta}}{D_{Red}} \tag{1.45}$$

La solución general de la ecuación diferencial 1.45 es:

$$\tilde{\theta} = A \cdot e^{y \cdot \sqrt{\frac{j \cdot \omega}{D_{Red}}}} + B \cdot e^{-y \cdot \sqrt{\frac{j \cdot \omega}{D_{Red}}}}; \quad A, B \in \mathbb{R} \tag{1.46}$$

Aplicando las condiciones de contorno 1.40 y 1.41, se obtiene la solución:

$$\tilde{\theta} = e^{-y \cdot \sqrt{\frac{j \cdot \omega}{D_{Red}}}} \tag{1.47}$$

Por lo tanto, el término $\frac{-1}{\tilde{\theta}'}$ es:

$$\frac{-1}{\tilde{\theta}'} = \frac{1}{\sqrt{\frac{j \cdot \omega}{D_{Red}}}} \tag{1.48}$$

Finalmente, la sustitución de 1.48 en 1.39 da la expresión de la impedancia de Warburg ($\tilde{Z_W}$):

$$\tilde{Z} = \frac{\tilde{E}}{\tilde{i}} = \overbrace{\frac{1}{k \cdot \bar{C}_{Red}|_e \cdot b \cdot e^{b \cdot \bar{E}}}}^{\tilde{Z}_f} + \overbrace{\frac{1}{n \cdot F \cdot D_{Red} \cdot \bar{C}_{Red}|_e \cdot b} \cdot \frac{1}{\sqrt{\frac{j \cdot \omega}{D_{Red}}}}}^{\tilde{Z}_W} \tag{1.49}$$

En la ecuación 1.49 se explicita el término $\frac{-1}{\tilde{\theta}'}$ para el caso de una capa límite infinita, lo que permite obtener la expresión correspondiente para la impedancia de Warburg (Z_W).

La impedancia de Warburg depende de la frecuencia angular de excitación (ω), siendo muy notoria a bajas frecuencias y despreciable para altas frecuencias. Esto es debido a que una oscilación rápida de voltaje repercute en una oscilación rápida de transferencia de carga, lo que permite a la transferencia de materia suplir la falta de reactivo en los periodos de transferencia de carga lenta. Sin embargo, en una oscilación lenta la transferencia de carga es rápida en periodos más largos de tiempo, impidiendo que la transferencia de materia aporte la cantidad necesaria de reactivo.

Impedancia de la doble capa

Además de las impedancias concernientes a las reacciones electroquímicas farádicas, existen impedancias asociadas a fenómenos no farádicos. Este es el caso de la capacitancia de la doble capa, que esta asociada a la polarización en las inmediaciones del electrodo.

La polarización del electrodo es un fenómeno que puede darse incluso sin flujo de corriente. Por ejemplo, la introducción de un metal en una disolución de yoduro potásico (KI) en agua, dará lugar a una capacitancia de la doble capa, ya que, la afinidad por el metal es diferente para el yoduro y el potasio.

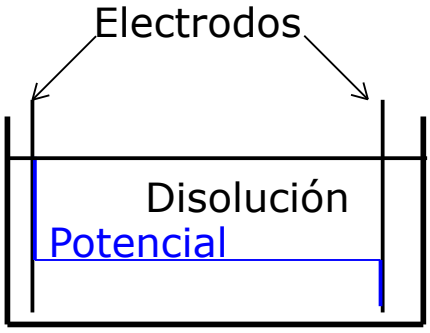


Figura 1.7: Distribución de voltaje para electrodos idealmente polarizables. Adaptado de Newman y Balsara 2021.

Supóngase que el yoduro tiene una afinidad muy superior a la del potasio por el metal, esto provocaría la interacción de parte del yoduro con la superficie del metal y, por tanto, un exceso de potasio en la disolución (i.e. desbalance de cargas en el electrolito).

Sin embargo, la fuerza eléctrica impide dicha situación, ya que, los iones yoduro interaccionando con la superficie del electrodo, están rodeados por iones potasio que compensan la carga eléctrica, formando así una capacitancia de la doble capa (Newman y Balsara 2021).

La formación de una capacitancia de la doble capa también permite explicar la caída de voltaje en los sistemas electroquímicos. Para el caso de la aplicación de un voltaje muy cercano al equilibrio, el flujo de corriente tiende a cero y, por lo tanto, la caída de voltaje no se debe al electrolito ni a reacciones electroquímicas, lo que impone una caída brusca de voltaje en las inmediaciones del metal, como se muestra en la figura 1.7.

Por lo tanto, el comportamiento de la impedancia de la doble capa es análoga a un elemento capacitivo. Al principio, la impedancia de la doble capa permitirá el paso de corriente hasta polarizar completamente el electrodo, lo que es análogo a la carga de un elemento capacitivo hasta su máxima capacidad. Tras ello, el paso de corriente se anula manteniendo la distribución de potencial, lo que es análogo al comportamiento de un elemento capacitivo que ha alcanzado su máxima carga.

Impedancias de un CPE

Los elementos de fase constante (CPE) están asociados a heterogeneidades en el electrodo (bordes de grano, distribuciones heterogéneas de voltaje, capas de óxido, distribuciones heterogéneas de materia, etc).Por ejemplo, un electrodo de disco rotatorio polarizable puede tener una distribución de impedancias dependiente con la posición, como se muestra en la figura 1.8.

En el caso de la figura 1.8 (i.e. disposición en paralelo), la inversa de la impedancia global ($\frac{1}{\tilde{Z}}$) puede ser calculada como la suma de la inversa de las impedancias locales ($\frac{1}{\tilde{z}}$) a lo largo de toda el área (A):

$$\frac{1}{\tilde{Z}} = \int_A \frac{1}{\tilde{z}} \cdot dA \tag{1.50}$$

Expandiendo el término $\tilde{z}$ en función de lo mostrado en la figura 1.8, se obtiene:

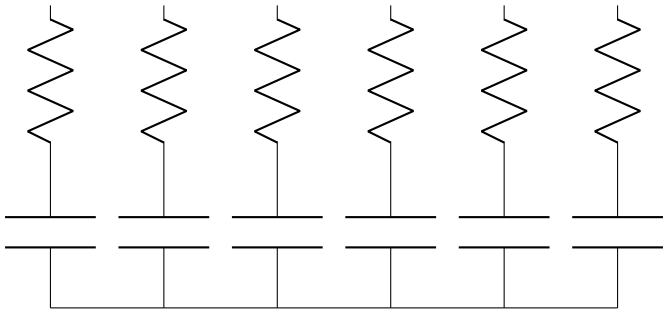


Figura 1.8: Distribución de impedancias locales a lo largo de un electrodo de disco rotatorio polarizable. Adaptado de Orazem y Tribollet 2008.

$$\frac{1}{\tilde{z}} = \sum_{i}^{\infty} \frac{1}{R_e + \frac{1}{j \cdot \omega \cdot C}} \tag{1.51}$$

Operando en la ecuación 1.51, se obtiene:

$$\frac{1}{\tilde{z}} = \sum_{i}^{\infty} \frac{j \cdot \omega \cdot C}{j \cdot \omega \cdot C \cdot R_e + 1} \tag{1.52}$$

Las constantes C_i y R_{ei} dependen de la posición. Por ello, se definen las constantes de tiempo τ_i:

$$\frac{1}{\tilde{z}} = \sum_{i}^{\infty} \frac{j \cdot \omega \cdot C}{j \cdot \omega \cdot \tau_i + 1} \tag{1.53}$$

Reorganizando términos en la ecuación 1.53, se obtiene:

$$\frac{1}{\tilde{z}} = \frac{1}{R_e} \cdot \sum_{i}^{\infty} 1 - \frac{1}{1 + j \cdot \omega \cdot \tau_i} \tag{1.54}$$

Suponiendo que la distribución de constantes de tiempo τ_i viene dada por la función de distribución G, se obtiene:

$$\frac{1}{\tilde{z}} = \frac{1}{R_e} \cdot \left(1 - \int_0^{\infty} \frac{1}{1 + j \cdot \omega \cdot \tau} \cdot G \cdot d\tau\right) \tag{1.55}$$

En caso que la función de distribución (G) sea una normal:

$$G(\tau) = \frac{1}{2 \cdot \pi \cdot \tau} \cdot \frac{sin(\alpha^* \cdot \pi)}{cosh\left((1 - \alpha^*) \cdot ln\left(\frac{\tau}{\tau_0}\right)\right) - cos(\alpha \cdot \pi)} \tag{1.56}$$

La impedancia de un CPE queda:

$$\tilde{Z} = R_e + \overbrace{\frac{1}{(j \cdot \omega)^{\alpha^*} \cdot Q}}^{\tilde{Z}_{CPE}} \tag{1.57}$$

Siendo α^* y Q parámetros ajustables independientes de la frecuencia.

Este elemento es extremadamente versátil, ya que, en función del valor que tome α^* pueden representar un elemento capacitivo ($\alpha^* = 1$), un elemento inductivo ($\alpha^* = -1$) o un elemento que no pueda ser clasificado estrictamente como capacitivo o inductivo ($\alpha^* \neq 1$ y $\alpha^* \neq -1$).

1.5 Representaciones gráficas en Espectroscopía de Impedancias Electroquímicas

En la sección 1.4.1, se han presentado los elementos más comunes en el ámbito de la Espectroscopía de Impedancias Electroquímicas por separado. Sin embargo, los sistemas electroquímicos reales están formados por una combinación de estos elementos. Por ello hay que utilizar métodos capaces de extraer información a partir del comportamiento agregado del sistema.

El primer paso en el análisis de cualquier espectro EIS es su representación. Las representaciones más habituales son los diagramas de Nyquist y la de Bode. Por un lado, los diagramas de Nyquist se basan en la representación de la impedancia a diferentes frecuencias en el plano complejo. La popularidad de este formato reside en la capacidad para identificar de forma sencilla procesos de activación (formación de semicírculos), influencia de la transferencia de materia (línea a 45°) y semicírculos cuyo centro esta por debajo del eje x (necesidad de un modelo más detallado). Sin embargo, la utilización de los diagramas de Nyquist enmascara la variable frecuencia, siendo esta desventaja aminorada mediante el etiquetado de la frecuencia en los puntos más significativos (Orazem y Tribollet 2008).

Por otro lado, los diagramas de Bode se basan en la representación del módulo de la impedancia y el ángulo de desfase frente a la frecuencia. Las ventajas de este formato son: mostrar explícitamente la frecuencia y la fácil identificación de elementos CPE (línea recta en el diagrama del módulo para altas frecuencias, cuya pendiente es $-\alpha^*$). Sin embargo, para frecuencias elevadas la resistencia del electrolito empaña los resultados del electrodo y la interfase. Este problema se soluciona mediante la resta de la resistencia del electrolito a la parte real de la impedancia, originando así los diagramas de Bode corregidos. Sin embargo, la estimación del valor exacto de la resistencia asociada al electrolito no suele ser sencilla, lo que dificulta la realización de estos diagramas (Orazem y Tribollet 2008).

Electrodo polarizable sin reacción química

El circuito eléctrico equivalente de un electrodo polarizable sin reacción electroquímica corresponde con el mostrado en la figura 1.9 y, cuya impedancia global se muestra en la expresión 1.58.

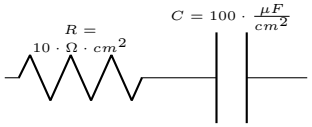


Figura 1.9: Electrodo polarizable sin reacción electroquímica. Adaptado de Orazem y Tribollet 2008

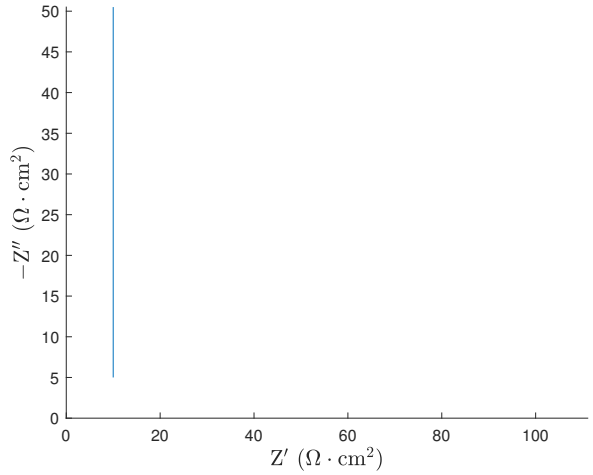
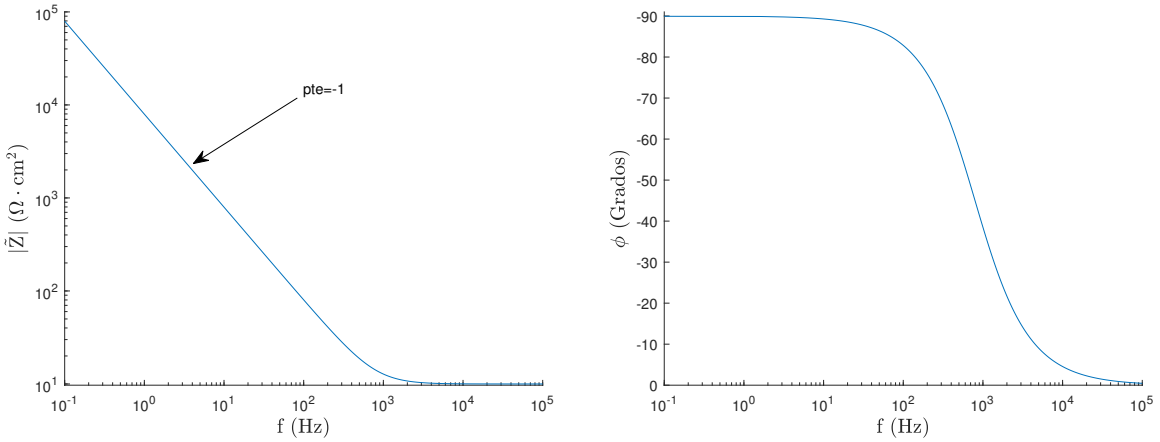


Figura 1.10: Diagrama de Nyquist para el electrodo polarizable sin reacción electroquímica mostrado en la figura 1.9.



(a) Módulo. **(b)** Desfase.

Figura 1.11: Diagrama de Bode para un electrodo polarizable sin reacción electroquímica.

$$\tilde{Z} = R - \frac{j}{\omega \cdot C} \tag{1.58}$$

Por un lado, el diagrama de Nyquist presentado en la figura 1.10 muestra una línea vertical, ya que, la parte real de la impedancia es independiente de la frecuencia y, es equivalente a la resistencia del electrodo, como se puede desprender la ecuación 1.58.

Por otro lado, el diagrama de Bode del módulo de la impedancia presentado en la figura 1.11a muestra como para frecuencias bajas el módulo de la impedancia sigue una línea recta con respecto a la frecuencia con pendiente -1. El valor de la pendiente es igual a $-\alpha^*$ y, por lo tanto, una pendiente de -1 indica la presencia de un elemento capacitivo puro. Sin embargo, para frecuencias altas la presencia del electrolito difumina el comportamiento lineal.

En cuanto al desfase, la figura 1.11b muestra un desfase de $-90°$ para frecuencias bajas, ocasionado por la acción del elemento capacitivo. Sin embargo, al aumentar la frecuencia la impedancia del elemento capacitivo se atenúa, lo que permite a la impedancia del electrolito dominar la respuesta, y por lo tanto, se mide un desfase de $0°$.

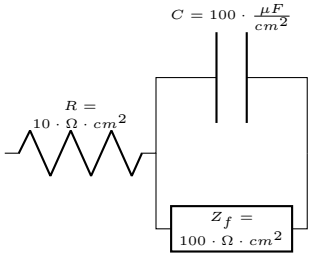


Figura 1.12: Electrodo polarizable con reacción electroquímica. Adaptado de Orazem y Tribollet 2008.

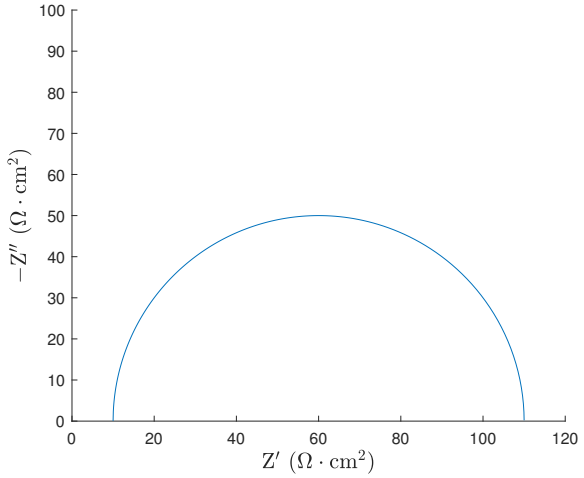


Figura 1.13: Diagrama de Nyquist para el electrodo polarizable con reacción electroquímica mostrado en la figura 1.12.

Electrodo polarizable con reacción electroquímica

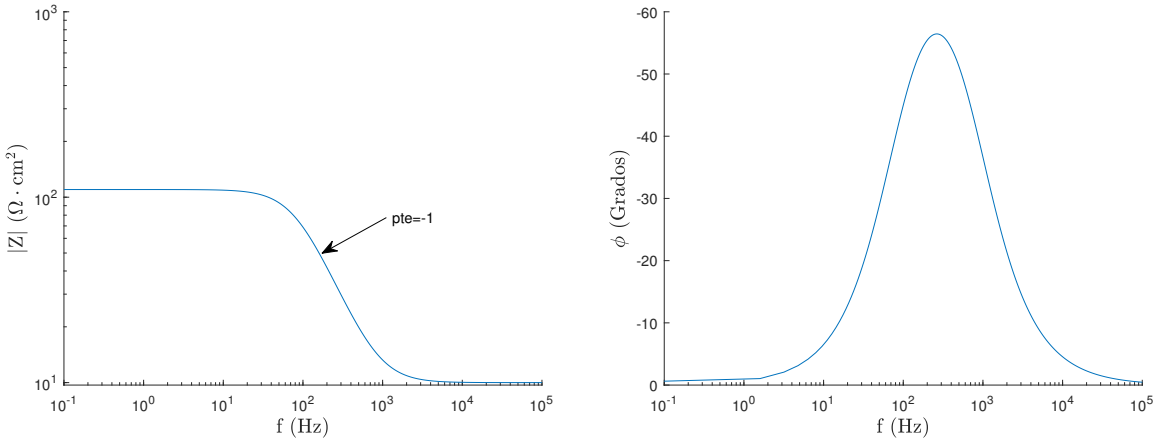
El circuito eléctrico equivalente de un electrodo polarizable con reacción electroquímica corresponde con el mostrado en la figura 1.12, cuya impedancia global se muestra en la expresión 1.59.

$$\tilde{Z} = R + \frac{\tilde{Z}_f}{(C \cdot \omega \cdot \tilde{Z}_f)^2 + 1} - \frac{\tilde{Z}_f^2 \cdot C \cdot \omega}{(C \cdot \omega \cdot \tilde{Z}_f)^2 + 1} \cdot j \tag{1.59}$$

Por un lado, el diagrama de Nyquist presentado en la figura 1.13 muestra un semicírculo, lo que indica la existencia de un proceso de activación, siendo en este caso producido por la disposición en paralelo de los elementos asociados a la reacción electroquímica y el elemento capacitivo de doble capa.

Además, permite obtener los valores de impedancia del electrolito y de la reacción electroquímica, mediante las intersecciones de la semicircunferencia con el eje real. El primer y segundo punto de intersección (i.e. $10\,\Omega \cdot \mathrm{cm}^2$ y $110\,\Omega \cdot \mathrm{cm}^2$, respectivamente) corresponden con la impedancia del electrolito, y la suma de la impedancia del electrolito e impedancia farádica, respectivamente.

Por otro lado, en el diagrama de Bode del módulo de la impedancia presentado en la figura 1.14a, para frecuencias bajas, el módulo de la impedancia ($|\tilde{Z}|$) tiende a la suma de las impedancias



(a) Módulo. **(b)** Desfase.

Figura 1.14: Diagrama de Bode para un electrodo polarizable con reacción electroquímica.

del electrolito y farádica, ya que, la impedancia del elemento capacitivo para estas frecuencias es muy elevada y, la corriente no pasa a través de él.

Para frecuencias intermedias se observa una línea recta con pendiente -1, lo que indica la presencia de un elemento capacitivo puro (como en el caso del electrodo polarizable sin reacción química).

Para frecuencias elevadas, el módulo de la impedancia tiende al valor de la impedancia del electrolito, ya que, la impedancia del elemento capacitivo tiende a cero y, por lo tanto, la corriente no fluye a través de la impedancia farádica.

En cuanto al desfase, la figura 1.14b muestra para frecuencias bajas y altas un desfase de 0°. Puesto que, para frecuencias bajas la corriente no pasa por el elemento capacitivo y, para frecuencias altas la resistencia del electrolito empaña el efecto del elemento capacitivo. Sin embargo, para frecuencias intermedias la impedancia del elemento capacitivo es relevante, lo que produce un desfase diferente de 0°.

1.5.1 Metodologías de interpretación

Existen 2 metodologías de interpretación diferentes de datos en Espectroscopía de Impedancias Electroquímicas, una basada en el desarrollo de ecuaciones particulares para cada sistema y, otra basada en circuitos eléctricos equivalentes.

La primera metodología utiliza una descripción matemática del sistema electroquímico estudiado. Sin embargo, implica una descripción matemática del sistema estudiado que, en función del nivel de detalle requerido conduce a desarrollos complejos y, por lo tanto, no siempre puede ser aplicada.

La segunda metodología, utiliza datos EIS para obtener un circuito eléctrico equivalente y el valor de sus impedancias. Una forma de llevarlo acabo es mediante la identificación de patrones en representaciones gráficas, como las comentadas en la sección 1.5, y posteriormente, ajustar el valor de las impedancias del circuito. Sin embargo, los valores de las impedancias no pueden ser relacionados con las constantes del sistema, ya que, provienen de un ajuste experimental y no del modelado matemático del sistema.

1.6 Proyecto genoma electroquímico

La identificación de patrones en espectros EIS es un aspecto clave para el estudio de los sistemas electroquímicos. Aunque, la forma de los espectros mostrados en la sección 1.5 son fácilmente reconocibles por un humano, los sistemas más complejos presentan matices más sutiles. Además, los espectros estudiados pueden tener un buen ajuste experimental a dos o más circuitos equivalentes, lo que complica de forma sustancial la tarea de identificación del circuito adecuado.

Como respuesta a esta dificultad, Digby D. Macdonald (Macdonald 2006) propuso la creación del proyecto genoma electroquímico. Dicho proyecto recibe su nombre por la similitud de patrones y espectros electroquímicos a genes y genomas, respectivamente.

El objetivo de este proyecto es la creación de una base de datos extensa, que albergue la mayor cantidad y diversidad de espectros electroquímicos posible. Ello unido a una inteligencia artificial capaz de detectar patrones de un espectro dado, y cotejarlos con los existentes en la base de datos, supondría un claro avance en términos de rapidez y sistematización en la proposición de circuitos eléctricos equivalentes.

Sin embargo, un proyecto de estas características requiere de la creación e identificación de cientos de miles o incluso millones de espectros EIS, lo que supondría un elevado coste material y temporal. Otra aproximación es la creación de la base de datos con espectros EIS extraídos de artículos ya publicados, ya que, la cantidad de artículos que presentan espectros EIS es muy grande.

1.7 Definición e historia de la Inteligencia Artificial

El rápido avance de la Inteligencia Artificial (IA) dificulta una definición precisa de la misma. Sin embargo, de forma general se puede entender IA como, la capacidad de un ordenador para comportarse en algunos aspectos de forma similar a los humanos (Kok et al. 2009).

La historia de la IA comenzó a fraguarse en el imaginario colectivo con la idea del hombre máquina, un ser creado artificialmente con capacidades cognitivas comparables a las del ser humano. Prueba de ello son las obras de autores de ciencia ficción como Julio Verne e Isaac Asimov, estableciendo este último en su obra Runaround las icónicas tres leyes de la robótica (Buchanan 2005).

Esta fantasía comienza a materializarse en 1942 con el trabajo de Alan Turing, mediante la creación del primer ordenador electromecánico para desencriptar el código enigma. El otro gran aporte de Alan Turing a la IA fue en 1950, mediante un artículo donde describía el test de Turing, un método para testear la inteligencia de un computador (Haenlein y Kaplan 2019).

El siguiente gran avance ocurrió en 1956, propiciado por la conferencia de Dartmouth Summer Research Project on Artificial Intelligence (DSRPAI) (Haenlein y Kaplan 2019), donde se acuñó el término Inteligencia Artificial y reunió a los considerados padres de la inteligencia artificial, como Marvin Minsky, Ray Solomonoff, John MacCarthy, etc.

Las dos décadas posteriores estuvieron marcadas por la creación de la herramienta de procesamiento de lenguaje ELIZA, que era capaz de simular una conversación con un humano. Y la

creación del algoritmo The Logic Theorist, capaz de resolver problemas sencillos como las torres de Hanoi (Haenlein y Kaplan 2019).

Estos avances iniciales ocasionaron un optimismo excesivo, ya que, el concepto estaba claro pero, la falta de potencia computacional (falta de memoria y poder de procesamiento) impidió un desarrollo más profundo del campo (Anyoha 2017). La falta de resultados provocó el cuestionamiento de la alta inversión en Inteligencia Artificial, lo que originó la retirada de fondos en 1973 por parte de los gobiernos británico y estadounidense, dando lugar al invierno de la IA (Haenlein y Kaplan 2019).

La retirada de gran parte de fondos paralizó el desarrollo de la IA durante el resto de la década de los 70. Sin embargo, en los años 80 la popularización de nuevas técnicas como el Deep Learning, que permitía el aprendizaje a través de la experiencia, y la creación del proyecto Fifth Generation Computer Project (FGCP) por parte del gobierno Japonés, reiniciaron el desarrollo en IA (Anyoha 2017).

En los últimos 30 años, los avances en hardware han permitido alcanzar grandes metas. Por ejemplo, el programa Deep Blue derrotó a Kasparov al ajedrez en 1997 (Haenlein y Kaplan 2019). En 2015, AlphaGO derrotó al campeón mundial de GO (Haenlein y Kaplan 2019), siendo el GO un juego con más posibilidades de movimientos que el ajedrez y, por lo tanto, más complejo. Finalmente, el último gran avance se ha dado en el campo del procesamiento de texto con el lanzamiento de GPT-3 en 2020 (Floridi y Chiriatti 2020) y, su variante especializada en chatbot, chatGPT en 2022 (Lund y Wang 2023). La figura 1.15 resume los hitos más importantes del campo de la IA comentados.

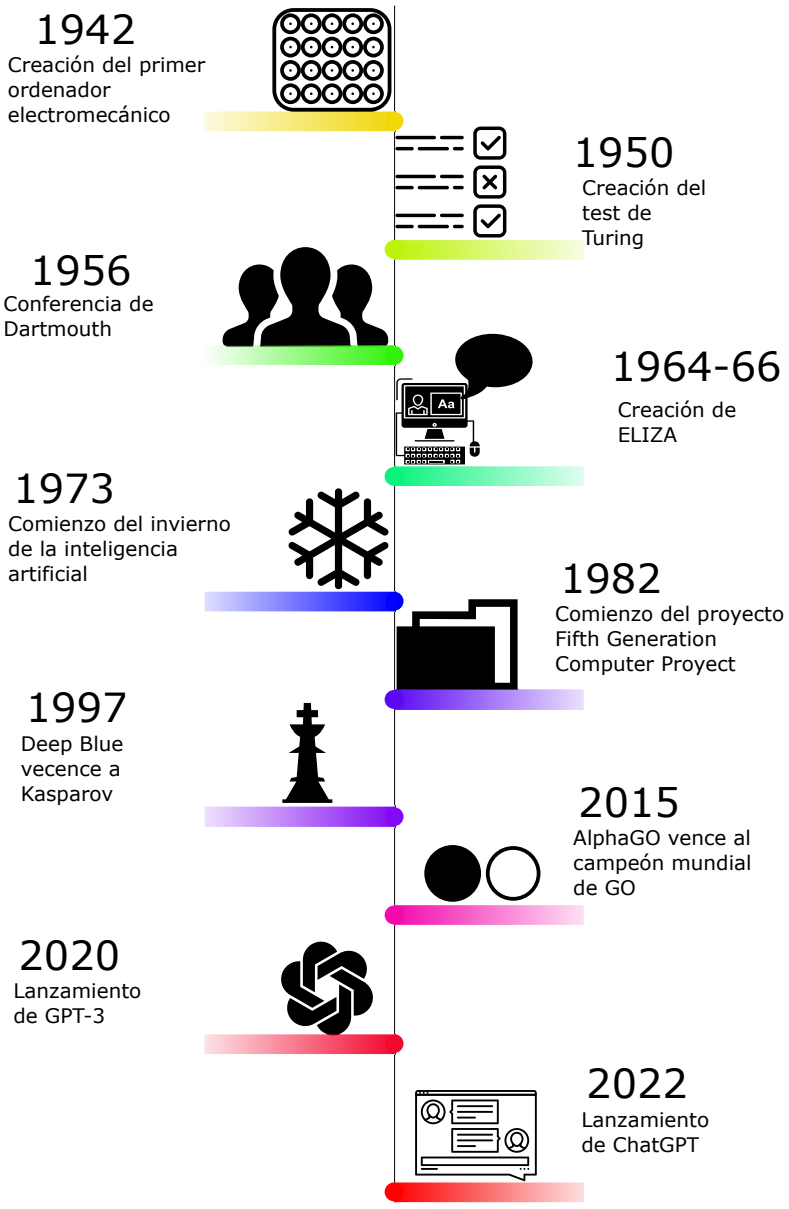


Figura 1.15: Línea temporal con los hitos más importantes de la inteligencia artificial

1.8 Aplicaciones de la inteligencia artificial

En la sección 1.7 y figura 1.15 se han presentado los ejemplos más icónicos de aplicaciones de la Inteligencia Artificial. Sin embargo, las aplicaciones de la IA no se restringen a las aplicaciones mostradas anteriormente, ya que, se pueden encontrar ejemplos de su uso en otros campos:

- Secuenciación del ADN: se han aplicado técnicas de Machine Learning para acelerar la secuenciación del ADN (Memeti y Pllana 2018), utilización de arquitecturas ANFIS (i.e. unión de lógica difusa con redes neuronales) para la identificación de genes (Dixit y Prajapati 2015), entre otros.
- Ecología: identificación de imágenes de animales mediante redes convolucionales (Tabak et al. 2019), estudio de la interacción entre diferentes especies mediante Deep Learning (Chen et al. 2016), entre otros.
- Electroquímica: detección de fallos en pilas de combustible PEM mediante técnicas EIS y Deep Learning (Lv et al. 2023), detección de la bacteria E. Coli mediante técnicas EIS mediante Support Vector Regression (Xu et al. 2020), entre otros.

La presencia de la inteligencia artificial en campos tan dispares se debe a su capacidad para encontrar patrones rápidamente, lo que es de utilidad en muchos ámbitos. Sin embargo, la IA requiere de un entrenamiento costoso en recursos materiales (datos) y temporales. Por ejemplo, el entrenamiento de GPT-3 ha supuesto el uso de 57×10^9 palabras y el ajuste de 175×10^9 parámetros (Tingiris y Kinsella 2021).

Capítulo 2

Objetivo, metodología y estructura

2.1 Objetivo

El objetivo de este Trabajo Final de Máster es la exploración y comparación del desempeño de diferentes algoritmos de Machine y Deep Learning en la tarea de recomendar circuitos eléctricos equivalentes para espectros de Espectroscopía de Impedancias Electroquímicas.

2.2 Metodología

En primer lugar, se realizó un proceso de depuración y expansión de una base de datos con espectros EIS etiquetados de acceso libre. En segundo lugar, se realizó un proceso de entrenamiento, testeo y optimización de hiperparámetros de diferentes algoritmos de Machine Learning y Deep Learning implementados en Matlab® con la base de datos depurada y expandida. Finalmente, se compararon entre sí los diferentes algoritmos de Machine Learning y Deep Learning con los hiperparámetros optimizados y, se seleccionó el que presentaba un desempeño mejor para la recomendación de espectros EIS.

2.3 Estructura

El presente Trabajo Final de Máster está estructurado en 6 capítulos descritos a continuación:

- En el primer capítulo, se realiza una introducción a la Espectroscopía de Impedancias Electroquímicas y a la Inteligencia Artificial.
- En el segundo capítulo, se describe el objetivo, metodología y estructura de este Trabajo Final de Máster.
- En el tercer capítulo, se introducen algunos conceptos fundamentales de Machine Learning y Deep Learning y, se describe el funcionamiento de los algoritmos utilizados.

- En el cuarto capítulo, se describe la base de datos de acceso libre empleada en este TFM y, las acciones efectuadas para su depuración y expansión.
- En el quinto capítulo, se describe la metodología de obtención de resultados e implementación de los diferentes algoritmos de Machine Learning y Deep Learning considerados.
- En el sexto capítulo, se muestran y analizan los resultados obtenidos del entrenamiento y testeo de los diferentes algoritmos considerados.
- En el séptimo capítulo, se presentan las conclusiones derivadas de los resultados presentes en el quinto capítulo.

Capítulo 3

Machine Learning y Deep Learning

3.1 Machine Learning

3.1.1 Definición y conceptos básicos

El Machine Learning es una disciplina dentro de las ciencias de la computación que se define como: Proceso de resolver un problema práctico mediante la recogida de datos y, la posterior construcción de un modelo estadístico basado en los datos recogidos (Burkov 2019).

El concepto de problema práctico engloba un número elevado de diferentes problemas. Sin embargo, los más habituales son los de regresión y clasificación. Un problema de regresión pretende hallar una relación continua entre los datos de entrada y salida (i.e. tomar entradas y asignarles una salida, estando ambas definidas en un conjunto continuo).

En cambio, un problema de clasificación consiste en la asignación de los datos de entrada a una clase en particular (i.e. tomar entradas definidas en un conjunto continuo y, asignarles una salida definida en un conjunto discreto). Este es el tipo de problema planteado en este Trabajo Final de Máster y, por lo tanto, a partir de aquí, la discusión sobre diferentes aproximaciones de entrenamiento y algoritmos hará referencia exclusivamente a problemas de clasificación.

En el ámbito del Machine Learning existen 4 paradigmas utilizados para la resolución de dichos problemas prácticos con modelos estadísticos:

- Aprendizaje supervisado: Los datos introducidos al modelo para su entrenamiento están etiquetados con la clase a la que pertenecen de antemano. El objetivo de este tipo de entrenamiento es que el modelo al finalizar el entrenamiento sea capaz de determinar la clase de un determinado input (Burkov 2019).
- Aprendizaje no supervisado: Los datos introducidos al modelo no están etiquetados de antemano, ya que, el objetivo final no es realizar una clasificación sino que, es obtener un vector transformado como salida. Este tipo de aprendizaje es utilizado para realizar reducción dimensional (i.e. el modelo es alimentado con un vector y este elimina los componentes que

aportan menos información, devolviendo un vector con menos dimensiones como resultado), clustering (i.e. agrupamiento de los datos de entrada en diferentes grupos en función de sus similitudes) (Burkov 2019).

- Aprendizaje semisupervisado: Algunos datos introducidos al modelo poseen etiqueta (una pequeña porción) y otros no (la mayoría). El objetivo de este tipo de aprendizaje es el mismo que en el aprendizaje supervisado (Burkov 2019).
- Aprendizaje por refuerzo: La inteligencia "vive" en un entorno del que recibe información. El objetivo es que la inteligencia artificial desarrolle la capacidad de dar lugar a acciones que le permitan alcanzar un objetivo final (Burkov 2019).

La utilización de una aproximación u otra está condicionado por el objetivo final (clasificación, reducción de clases, etc) y por las características de los datos usados como entrenamiento (datos etiquetados, parcialmente etiquetados o no etiquetados). En el caso de este Trabajo Final de Máster, se pretende recomendar circuitos eléctricos equivalentes para espectros EIS donde, los espectros están etiquetados de antemano en una base de datos. Por lo tanto, el tipo de aprendizaje idóneo para esta aplicación es el aprendizaje supervisado.

El funcionamiento general del aprendizaje supervisado es:

El primer elemento en el aprendizaje supervisado son las bases de datos que albergan numerosos ejemplos, estando cada ejemplo formado por la dupla características y etiqueta. Por un lado, las características son agrupadas en una matriz de características ($\mathbf{X}$) que tiene tantas filas y columnas como características y ejemplos haya, respectivamente. Por otro lado, las etiquetas que indican las clases a las que pertenecen los ejemplos son agrupadas en el vector de clases ($\vec{Y}$).

El segundo elemento es el planteamiento del problema como un problema de optimización. Para ello, es necesario definir una función objetivo a optimizar, lo que se denomina en Machine Learning como función de coste. Dicha función de coste será utiliza por un algoritmo de Machine Learning, que fijará la dependencia de la función de coste con los parámetros optimizables. Además, cada algoritmo de Machine Learning aporta parámetros no optimizables en la optimización de la función de coste y, que marcan el comportamiento del algoritmo en líneas generales, conocidos como hiperparámetros.

La resolución de este problema de optimización es conocido como entrenamiento y, permite que las IAs clasifiquen un vector de características de forma correcta en una clase o al menos, con un error inferior previo al entrenamiento.

Sin embargo, la resolución del problema de optimización de forma satisfactoria no asegura un buen desempeño de la IA en la clasificación, ya que, existe el fenómeno de sobreajuste (u overfitting en inglés). El sobreajuste se da cuando una IA es capaz de predecir con un alto grado de exactitud la clase para los datos de entrenamiento pero, tiene serias dificultades para clasificar datos con los que no ha sido entrenada, es decir, carece de la capacidad para generalizar los resultados a nuevos ejemplos. Por ello, todo proceso de entrenamiento debe ir acompañado de un proceso de test que permita verificar el desempeño de la IA con ejemplos no utilizados durante el entrenamiento.

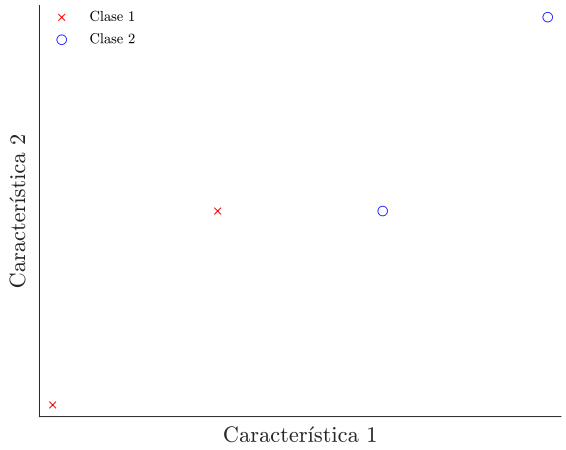


Figura 3.1: Representación de ejemplos en el espacio de características.

3.1.2 Algoritmos de Machine Learning

Los diferentes algoritmos de Machine Learning son diferentes aproximaciones estadísticas a la resolución de los problemas propios de este campo (clasificación, clusterización, etc) y, por lo tanto, la elección de un algoritmo u otro tiene un gran impacto en la solvencia de la IA tras el entrenamiento. Los algoritmos considerados en este Trabajo Final de Máster son: Support Vector Machine, Neural Network, Naive Bayes, K Nearest Neighbor y Decision Tree.

Support Vector Machine

El algoritmo de Support Vector Machine (SVM) interpreta cada ejemplo de la base de datos como un vector perteneciente a un hiperespacio abstracto. Dicho espacio se conoce como el espacio de características y, tiene tantas dimensiones como elementos posee el vector de características, como se muestra en la figura 3.1.

Una vez colocados los ejemplos en el espacio de características y, etiquetados con la clase a la que pertenecen (i.e. -1 para la clase 1 y 1 para la clase 2). El algoritmo SVM separa las diferentes clases, mediante el trazado de un hiperplano (Kowalczyk 2017). Un hiperplano es una región de $n-1$ dimensiones que divide un espacio n-dimensional en dos regiones diferentes. Por ejemplo, para dimensiones 1, 2 y 3, los hiperplanos correspondientes son el punto, la recta y el plano, cuyas dimensiones son 0, 1 y 2, respectivamente.

La ecuación de un hiperplano es la generalización de la ecuación de una recta a cualquier número de dimensiones:

$$\vec{w} \cdot \vec{\mathbf{X}}_i + n = 0 \tag{3.1}$$

La ecuación 3.1 muestra la ecuación de un hiperplano donde, $\vec{X}_i$ hace referencia al vector de características, $\vec{w}$ es el vector normal al hiperplano y, n es la intersección de hiperplano con los ejes.

La ecuación del hiperplano permite evaluar rápidamente la posición relativa de un punto respecto al hiperplano. Puesto que, el producto escalar de $\vec{w}$ con $\vec{\mathbf{X}}_i$ y, la posterior suma de n, dará un número positivo o negativo en función si el punto se encuentra en una región del espacio u otra

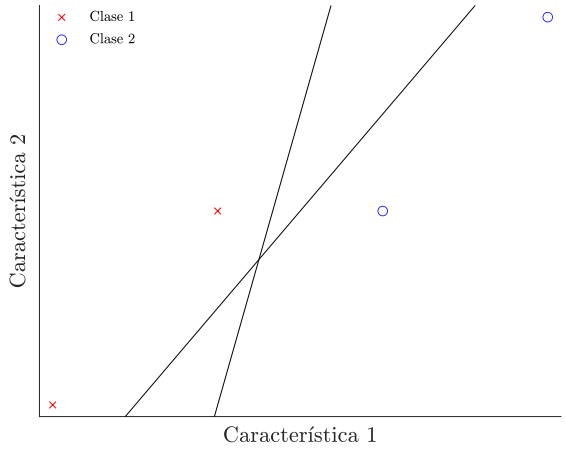


Figura 3.2: Hiperplanos dividiendo el espacio de características en 2 zonas, una para cada clase.

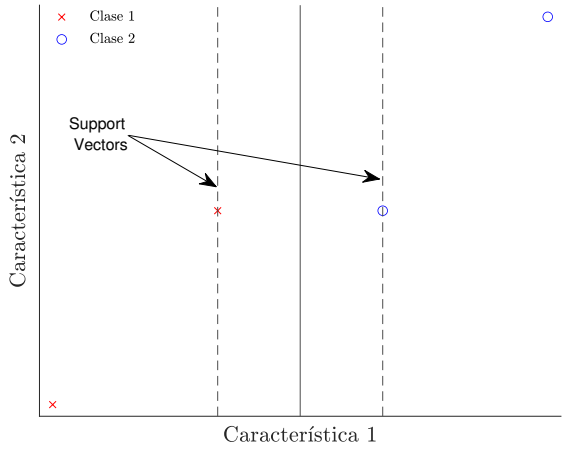


Figura 3.3: Support vectors asociados a un hiperplano

región del espacio (siendo cero si el punto esta sobre el hiperplano), lo que es de gran utilidad para la clasificación.

La figura 3.2 muestra como se puede realizar la clasificación en base a un hiperplano. Este divide el espacio en 2 zonas, cada una asociada a una clase. Sin embargo, en general, no existe un hiperplano clasificador único.

Por este motivo, se necesita de una métrica que permita comparar diferentes hiperplanos entre sí y, elegir el mejor. Dicha métrica es la distancia del hiperplano a los support vectors (i.e. a los puntos más cercanos del hiperplano), como se muestra en la figura 3.3, considerándose superior un hiperplano cuanto mayor sea esta distancia, ya que, aumenta la capacidad de generalización (Kowalczyk 2017). La distancia del hiperplano a los support vectors es inversamente proporcional al módulo del vector $\vec{w}$ y, por lo tanto, se puede plantear el siguiente problema de optimización:

$$\begin{aligned} \underset{\vec{w},\ n}{\text{Minimizar:}} \quad & ||\vec{w}|| \\ \text{Bajo la condición:} \quad & Y_i \cdot (\vec{w} \cdot \vec{\mathbf{X}}_i + n) \geq 1 \\ & \forall\ i\ \in\ \{1, 2, ..., i\} \end{aligned} \tag{3.2}$$

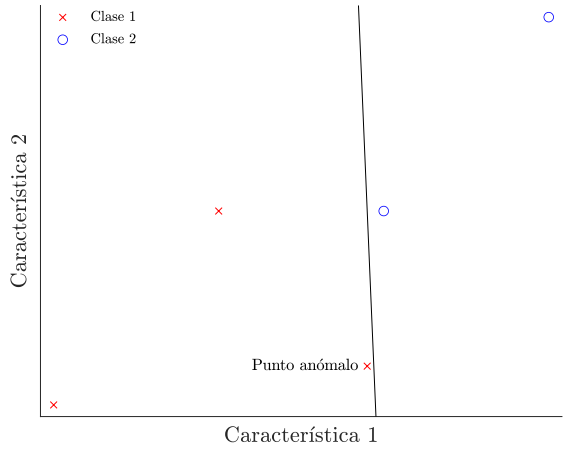


Figura 3.4: Influencia de los puntos anómalos en el trazado de hiperplanos.

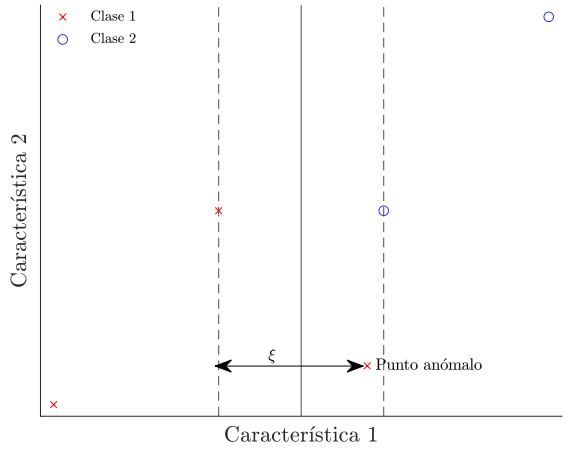


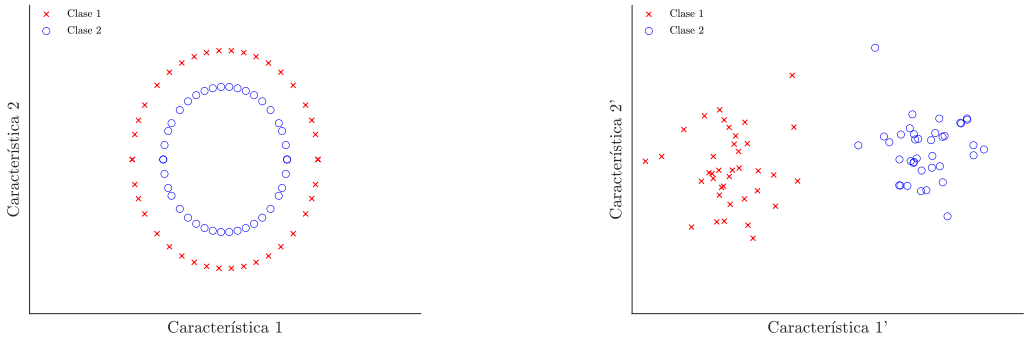
Figura 3.5: Efecto del margen suave en el trazado de hiperplanos en clases con puntos anómalos.

Otro problema es la clasificación de clases en las que existen puntos anómalos, ya que, el algoritmo intentará que el plano divida ambas clases sin puntos mal clasificados. Esto conlleva al trazado de un hiperplano sesgado por los puntos anómalos, como se muestra en la figura 3.4.

La solución para estos casos viene dada por la introducción de márgenes suaves, o soft margin en inglés, que permiten incurrir en clasificaciones erróneas (Kowalczyk 2017). Para ello, se introduce el vector $\vec{\xi}$ que, cuantifica la distancia entre una clasifcación errónea y su márgen correspondiente, como se muestra en la figura 3.5. Además, se modifica la función a minimizar para que penalice valores elevados en el vector $\vec{\xi}$:

$$\begin{aligned} \underset{\vec{w},\ n,\ \xi}{\text{Minimizar:}} \quad & ||\vec{w}|| + C' \cdot \sum_{i=1}^{m} \xi_i \\ \text{Bajo las condiciones:} \quad & Y_i \cdot (\vec{w} \cdot \vec{\mathbf{X}}_i + n) \geq 1 - \xi_i \\ & \xi_i \geq 0 \ \forall \ i \in \{1, 2, ..., i\} \end{aligned} \tag{3.3}$$

Donde C' es el Box Constraint, un hiperparámetro que regula la penalización de los errores cometidos. Los valores elevados de Box constraint asignan mucha importancia a los errores cometidos al trazar al hiperplano y, por lo tanto, un valor de Box Constraint próximo al infinito convier-



(a) Previo a la aplicación del Kernel **(b)** Posterior a la aplicación del Kernel

Figura 3.6: Aplicación de un Kernel a datos no linealmente separables.

te el problema de optimización 3.3 en el 3.2. En cambio, un valor de Box Constraint próximo a 0 permitirá un elevado número de errores, comprometiendo la capacidad de clasificación del algoritmo. Por lo tanto, la constante Box Constraint permite priorizar entre modelos con una elevada bondad de clasificación (valores elevados de Box Constraint), o modelos con una elevada capacidad de generalización (valores pequeños de Box Constraint).

Un problema adicional es que los datos no sean linealmente separables, es decir, que el trazado de un hiperplano no sea capaz de separar las diferentes clases, como se muestra en la figura 3.6a. Esto obliga a transformar el espacio de características original en uno modificado donde, las clases sean linealmente separables, como se muestra en la figura 3.6b.

Estas transformaciones son muy costosas computacionalmente, ya que, implican cambiar cada coordenada de cada ejemplo. Por ello, se utilizan los Kernels que realizan dichas transformaciones con un coste computacional moderado. Por defecto, el Kernel utilizado es el lineal que no transforma el espacio de características y solo es capaz de clasificar datos linealmente separables. Sin embargo, para datos no linealmente separables es posible utilizar Kernels polinómicos, gaussianos, etc.

Neural networks

Las redes neuronales están compuestas por capas interconectadas entre sí, como se muestra en la figura 3.7. Dichas capas pueden ser clasificadas atendiendo a su función en 3 tipos diferentes. Las capas de entrada, intermedias u ocultas, y de salida, cuyas funciones son captar información del exterior, procesar la información y comunicar la información al exterior, respectivamente.

El elemento fundamental de estas capas son las neuronas. En este contexto, las neuronas son unidades de procesamiento capaces de operar de forma paralela (Krose y Smagt 1996), cuyo algoritmo de funcionamiento es el siguiente:

- En primer lugar, una neurona (denotada por el subíndice j) recibe información de i fuentes de información (cada una denominada y_{input_i}) y agrega la información. Para ello, se pondera cada fuente de información con un peso o importancia ($\mathbf{W}$) y se suma. Además, se añade un término independiente (i.e no depende de las entradas de la neurona) denominado sesgo (b). Así pues, la información agregada de la neurona j (s_j) viene dada por:

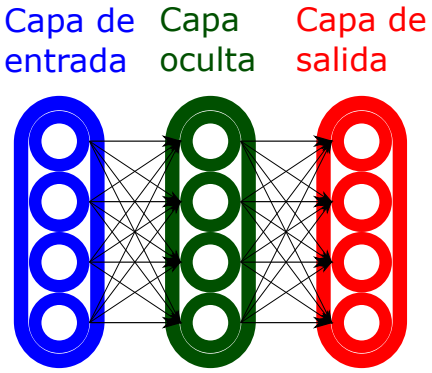
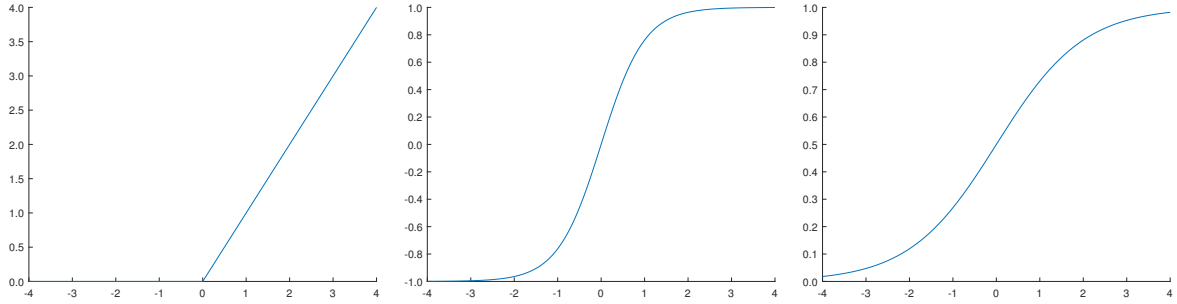


Figura 3.7: Disposición de las capas en una red neuronal.



(a) ReLU **(b)** Tangente hiperbólica **(c)** Sigmoide

Figura 3.8: Ejemplos de funciones de activación.

$$s_j = \vec{b}_j + \sum_{i=1}^{n} y_{input_i} \cdot \mathbf{W}_{ij} \tag{3.4}$$

En segundo lugar, la información agregada (s) es introducida como argumento de entrada a una función de activación:

$$y_{output_j} = f_{ac}(s_j) \tag{3.5}$$

Las funciones de activación son el elemento de la neurona que genera la salida de la neurona en función de la información agregada. Algunas de las funciones de activación más comunes se recogen en la figura 3.8.

- En tercer lugar, la neurona envía la salida de la función de activación (y_{output}) a las neuronas conectadas con su salida.

Teniendo esto en cuenta, se desprende que el número de neuronas en las capas de entrada y salida no pueden ser fijadas a voluntad, ya que, deben tener el número correcto de neuronas para interaccionar correctamente con el exterior. La capa de entrada debe tener tantas neuronas como fuentes del exterior haya (i.e. tantas como dimensiones tenga el espacio de características), ya que, cada neurona se encargará de captar la información de una única fuente de información. La capa de salida debe poseer tantas neuronas como clases diferentes haya, ya que, si la red clasifica correctamente, cada neurona de salida se activará preferentemente para una clase en concreto.

En cambio, las capas ocultas no presentan ninguna restricción en cuanto al número de neuronas, ya que, su comunicación es exclusivamente entre capas de entrada y salida.

Los hiperparámetros de este algoritmo son todas las variables que definen el comportamiento de las neuronas y la arquitectura de la red: las conexiones entre capas, el número de capas, el número de neuronas en cada capa oculta y la función de activación para cada capa de neuronas.

Naive Bayes

El algoritmo Naive Bayes (NB) recibe el nombre de Bayes por la utilización de la regla de Bayes y, el adjetivo naive (o ingenuo, en castellano) de las simplificaciones poco realistas que realiza. El funcionamiento del algoritmo esta basado en la regla de Bayes. Por ello, el objetivo del entrenamiento es la estimación de probabilidades que permitan predecir la clase de un ejemplo a partir de su vector de características.

Por ejemplo, si se tiene un set de entrenamiento que contiene ejemplos cuyos vectores de características $\vec{\mathbf{X}}_i$ tienen n elementos y cuyo vector clases $\vec{Y}_i$ tiene m posibles clases, se puede aplicar la regla de Bayes y obtener la probabilidad de pertenecer a la clase $\vec{Y}_k$ dada la característica $\mathbf{X}_{ij}$:

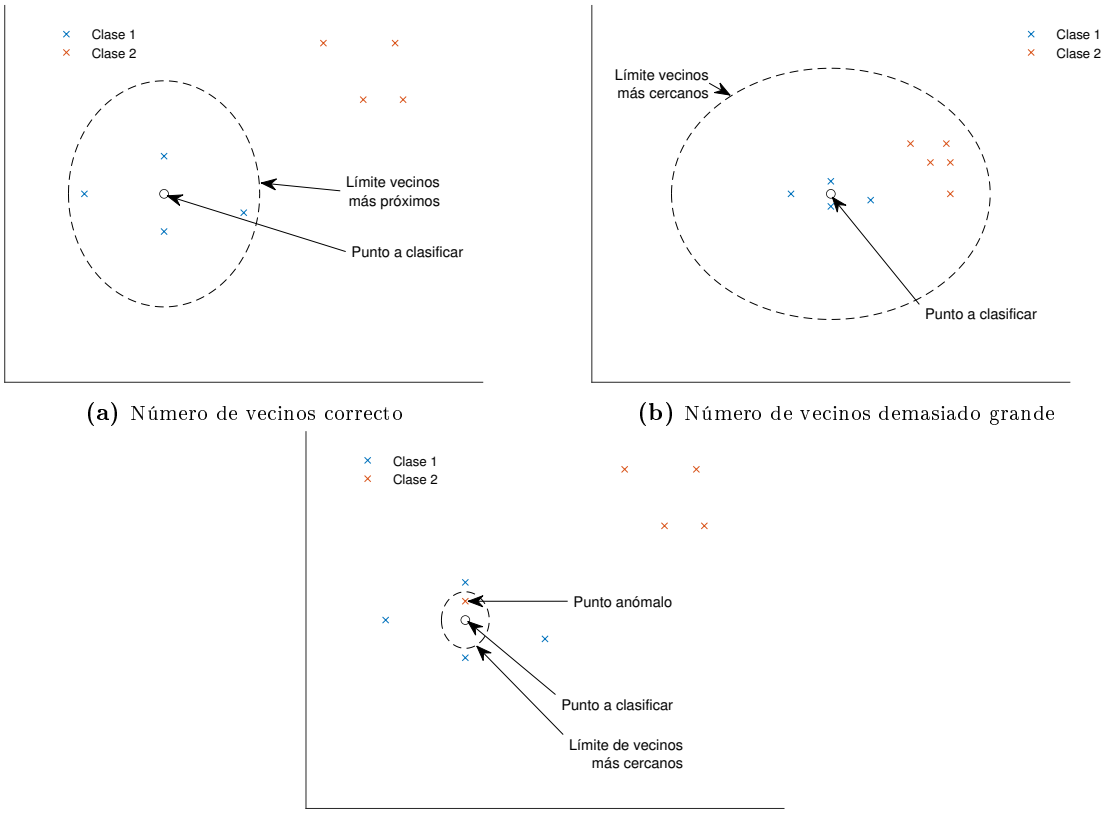
$$\mathbb{P}(\vec{Y}_k|\mathbf{X}_{ij}) = \frac{\mathbb{P}\mathbf{X}_{ij}|\vec{Y}_k) \cdot \mathbb{P}(\vec{Y}_k)}{\sum_m \mathbb{P}(\mathbf{X}_{ij}|\vec{Y}_m) \cdot \mathbb{P}(\vec{Y}_m)} \tag{3.6}$$

Aunque la ecuación 3.6 permite realizar predicciones a partir del vector de características, es computacionalmente muy costosa. Sin embargo, asumiendo que los elementos del vector de características ($\mathbf{X}_{ij}$) son condicionalmente independientes de la clase (lo que es poco razonable, y por lo ello, la denominación ingenuo o naive) es posible obtener:

$$\mathbb{P}(\vec{Y}_k|\vec{\mathbf{X}}_i) = \frac{\prod_{j=1}^{n} (\mathbb{P}(\mathbf{X}_{ij}|\vec{Y}_k)) \cdot \mathbb{P}(\vec{Y}_k)}{\sum_m \prod_{j=1}^{n} (\mathbb{P}(\mathbf{X}_{ij}|\vec{Y}_m)) \cdot \mathbb{P}(\vec{Y}_m)} \tag{3.7}$$

Por lo tanto, durante el entrenamiento se calculan las probabilidades $\mathbb{P}(\mathbf{X}_{ij}|\vec{Y}_m)$ y $\mathbb{P}(\vec{Y}_m)$ con los datos del set de entrenamiento y, para la clasificación, dado un vector de ejemplos $\vec{\mathbf{X}}_i$ se utilizará la ecuación 3.7 para realizar las predicciones.

Para el cálculo de estas probabilidades en características continuas es necesario suponer una distribución (normal, box, epanechnikov, etc) y, en base a esta estimar las probabilidades necesarias. Por lo tanto, el entrenamiento del algoritmo Naive Bayes es diferente al resto, ya que, no necesita minimizar una función de coste, simplemente se limita a actualizar las probabilidades necesarias según avanza el entrenamiento.



(a) Número de vecinos correcto

(b) Número de vecinos demasiado grande

(c) Número de vecinos demasiado pequeño

Figura 3.9: Efecto del número de vecinos en la clasificación de un punto para el algoritmo K Nearest Neighbor.

K Nearest Neighbor

El algoritmo K Nearest Neighbor (KNN) recibe su nombre por clasificar nuevos ejemplos en función de la clase de los k vecinos más próximos. Para ello, durante el entrenamiento almacena la información de los vectores de características y la clase para cada ejemplo. Tras ello, cada vez que se necesita realizar una predicción, el modelo evalúa la distancia entre cada ejemplo suministrado en el entrenamiento y el nuevo ejemplo, clasificándolo como la clase más común de sus k vecinos más próximos, como se muestra en la figura 3.9a.

Un aspecto fundamental para una clasificación correcta por parte del KNN es la elección del número de vecinos, ya que, no es un parámetro optimizable durante el entrenamiento y, por lo tanto, es un hiperparámetro.

Por un lado, la elección de un número de vecinos demasiado pequeño provoca overfitting, ya que, el algoritmo solo se fijará en los vecinos más próximos y, por lo tanto, será muy sensible al ruido (Steinbach y Tan 2009), como se muestra en la figura 3.9c. Por otro lado, la elección de un número de vecinos demasiado grande aumenta la posibilidad de tener en cuenta puntos de otras clases, lo que termina distorsionado las predicciones (Steinbach y Tan 2009), como se muestra en la figura 3.9b.

Otro aspecto importante es la medida de las distancias entre las observaciones (e.g. distancia de Minkowski, Euclídea, etc), ya que, la utilización de una métrica u otra condiciona la elección de

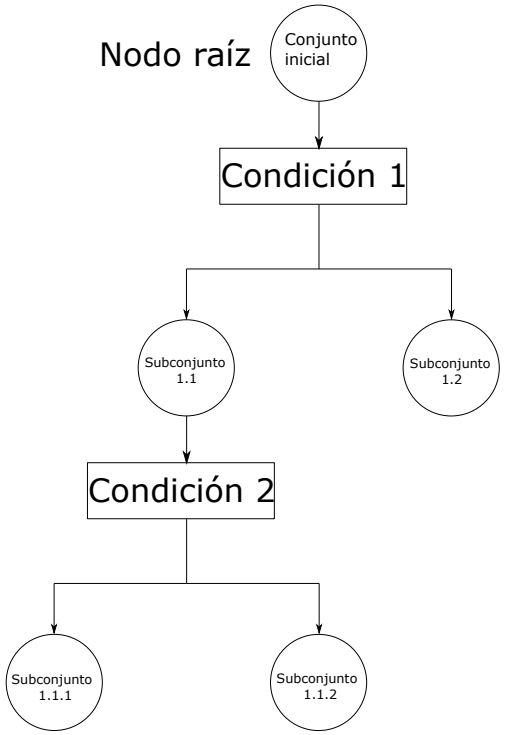


Figura 3.10: Estructura esquemática del Decision Tree.

los vecinos más próximos a la hora de realizar la clasificación. Esto es un parámetro que debe ser seleccionado de antemano y, por lo tanto, es otro hiperparámetro.

Decision Tree

El algoritmo Decision Tree (DT) consiste en la creación de un árbol de decisión. Para ello, se parte de un nodo raíz (i.e. un nodo que contiene todas las observaciones) y el algoritmo DT busca la condición más adecuada para separar el conjunto inicial de observaciones en subconjuntos lo más homogéneos posibles (i.e. que cada nodo tenga la menor variedad de clases posible). Los subconjuntos resultantes residen en nodos que pueden ser divididos del mismo modo que el nodo raíz, como se muestra en la figura 3.10.

El algoritmo DT posee un único hiperparámetro, el número máximo de divisiones permitidas. Un número elevado de divisiones permite la clasificación de problemas más complejos puesto que, se tendrán más condiciones en cuenta a la hora de generar los subconjuntos finales. Sin embargo, un exceso de divisiones puede generar modelos demasiado sensibles al ruido en las observaciones, como se observa en la figura 3.11.

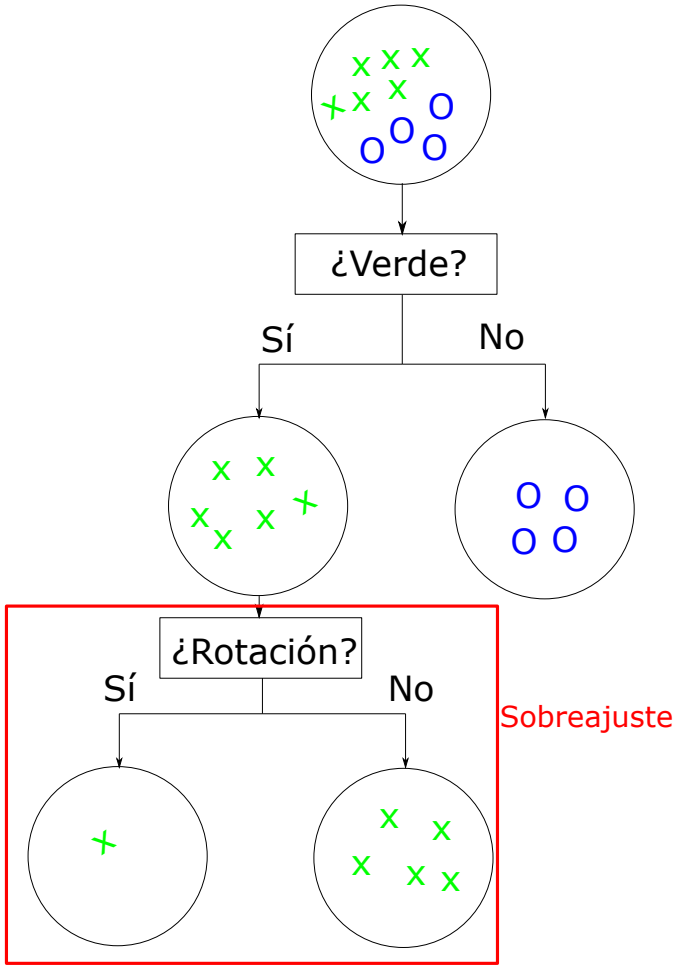


Figura 3.11: Sobreajuste provocado por demasiadas divisiones en el algoritmo decisión tree.

3.2 Deep Learning

Los algoritmos de Deep Learning son un subconjunto de algoritmos de Machine Learning que representan la realidad de forma jerárquica, es decir, construyen conceptos complejos en base a conceptos más simples (Goodfellow 2016). Por lo tanto, se suprime la necesidad de buscar unos parámetros de entrada adecuados para el aprendizaje (como sucede en el caso del Machine Learning) y, el propio algoritmo busca o genera las entradas más adecuadas de entrada para el aprendizaje.

Existen diversos algoritmos de Deep Learning (e.g. redes neuronales recurrentes, autoencoders, redes neuronales con más de 3 capas, redes convolucionales, etc). Sin embargo, en este Trabajo Final de Máster se consideran únicamente las redes convolucionales, cuyos fundamentos se explican a continuación.

3.2.1 Convolutional Neural Networks

Las redes convolucionales, o Convolutional Neural Networks (CNN) en inglés son un tipo de red neuronal especializadas en tratar datos tipo matriz, como las imágenes. Estas están formadas por 2 partes (parte convolucional y neuronal tradicional), que trabajando de forma conjunta son capaces de identificar patrones clave (parte convolucional) y, clasificar imágenes en base a estos (parte neuronal).

La parte convolucional utiliza la operación matemática convolución, cuya definición matemática es:

$$(f * g)(t) = \int f(a) \cdot w(t - a) \cdot da \tag{3.8}$$

Una imagen está compuesta por elementos discretos denominados píxeles, ordenados en 2 dimensiones (i.e. altura y anchura). Por ello, es necesario discretizar y expandir a dos dimensiones la definición 3.8 para que sea aplicable a una imagen:

$$(f * g)(i, j) = \sum_m \sum_n f(m, n) \cdot g(i - m, j - n) \tag{3.9}$$

El operador convolución necesita ser aplicado a dos funciones, como se muestra en las expresiones 3.8 y 3.9. Por ello, se necesita de una función auxiliar denominada Kernel o filtro, como se muestra en la figura 3.12.

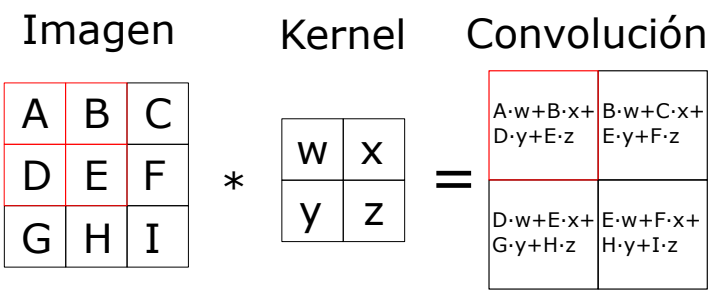
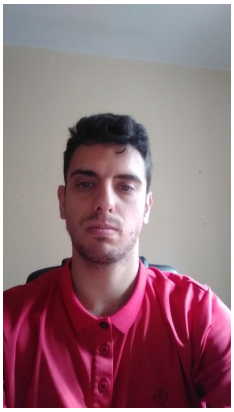


Figura 3.12: Convolución de una imagen. Adaptado de Goodfellow 2016.



(a) Imagen original.

(b) Imagen tras aplicar un Kernel.

Figura 3.13: Aplicación de un Kernel (especializado en la detección de bordes) a una imagen.

Estos Kernels son el motivo principal de aplicar el operador convolución a las imágenes, ya que, un Kernel adecuado puede extraer información sobre patrones de la imagen (e.g. reconocimiento de bordes, objetos, etc) para su posterior clasificación, como se muestra en la figura 3.13.

Esta extracción de patrones claves de la información de entrada se realiza en las etapas convolucionales, que están formadas por 3 capas principales:

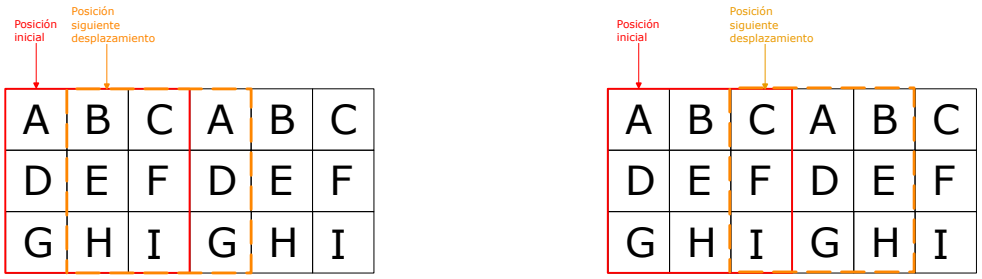
El primer elemento de la etapa convolucional es la capa convolucional que acepta datos externos a esta (i.e. de otras etapas o de la imagen) y, les aplica diferentes filtros para la extracción de patrones.

Los hiperparámetros que definen este elemento son:

- Número de filtros: Establece el número de filtros que serán aplicados a los datos de entrada.
- Tamaño del filtro: Establece las dimensiones del filtro a ser aplicado y, por lo tanto, modula el número de píxeles involucrados en la detección de patrones.
- Stride: Establece el número de píxeles que se desplaza el filtro por la imagen y, por lo tanto, un stride elevado provocará una gran compresión de la información de entrada, como se muestra en la figura 3.14.
- Padding: Establece el número de píxeles extra en los bordes de los datos de entrada, como se muestra en la figura 3.15, y, por lo tanto, controla la compresión de la información de entrada. En Matlab® existe la opción de fijar el padding a "Same", lo que condiciona el comportamiento del padding con el valor del stride del siguiente modo: Si el stride tiene un valor de 1x1, el padding se fijará en un valor que impida la compresión de la información de entrada. En otro caso, la información de salida tendrá la dimensión ceil (Dimensión datos de entrada/stride).

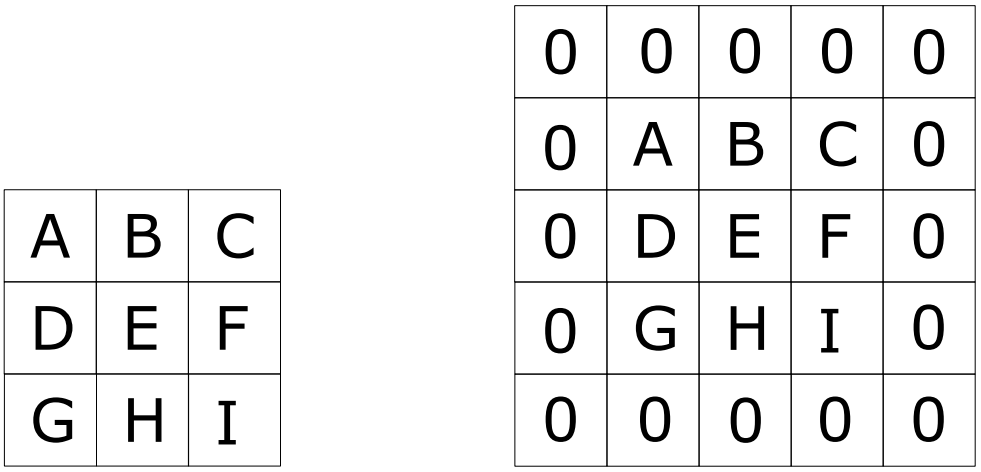
Los parámetros a optimizar durante el entrenamiento en la capa convolucional son los valores peso y sesgo para cada Kernel utilizado.

El segundo elemento de la etapa convolucional es la capa de función de activación que permite detectar los patrones clave de cada canal. Estas funciones de activación son las mismas que las utilizadas en las redes neuronales tradicionales (e.g. sigmoide, tangente hiperbólica, ReLU, etc).



(a) Stride 1. **(b)** Stride 2.

Figura 3.14: Efecto de parámetro stride sobre la convolución.



(a) Imagen original. **(b)** Imagen tras aplicar padding.

Figura 3.15: Efecto de aplicar un padding de 1 en todas las direcciones a una imagen.

El tercer elemento de la etapa convolucional es el pooling que realiza un resumen estadístico de la información proveniente de la función de activación, como puede ser la media, elección del valor más alto, etc. El objetivo de esta capa es doble: Por un lado, proporciona a la red convolucional la capacidad de reconocer patrones de forma invariante (i.e. reconocer patrones incluso si estos están desplazados o rotados). Por otro lado, realizar un resumen estadístico implica que la salida de la capa de pooling contiene menos información que la entrada y, por lo tanto, ayuda a reducir el tiempo de cómputo en las capas posteriores (Goodfellow 2016).

Las características que gobiernan esta capa son el pool size (que es el equivalente al tamaño del filtro en la parte convolucional), stride y padding, que funcionan del mismo modo que en la capa convolucional.

Además, es posible encontrar una cuarta capa (situada entre la capa de funciones de activación y la de pooling) encargada de realizar una normalización antes de realizar el resumen estadístico. El objetivo de esta capa es evitar que algunas variables se salgan de rango y, su gradiente sea prácticamente 0, lo que provoca una ralentización de la convergencia (Ioffe y Szegedy 2015).

Las características que gobiernan esta capa son el ratio de aprendizaje (cuyo funcionamiento es idéntico al explicado para la capa convolucional), y los parámetros necesarios para realizar la normalización de cada canal (un valor de escala y término independiente).

Tras pasar por varias etapas convolucionales, la información llega a la segunda parte de la red convolucional (i.e. la red neuronal tradicional), donde se realiza la clasificación. El número de capas que contiene la etapa neuronal tradicional no esta definido, ya que, se pueden poner capas de redes neuronales de forma indefinida. Sin embargo, el número mínimo de capas neuronales son 2. La primera, sirve como nodos de entrada a la información procedente de las etapas convolucionales y, por lo tanto, tendrá tantas neuronas como fuentes de información tenga la última etapa convolucional. La segunda, transmite el resultado de la clasificación al exterior y, por lo tanto, tendrá tantas neuronas como clases distintas haya. El hiperparámetro asociado a esta capa neuronal es la función de activación, y los parámetros optimizables son los pesos y sesgos.

En cuanto al flujo de información en las redes convolucionales está organizado en tensores de rango 3, es decir, matrices de 3 dimensiones donde, a la tercera dimensión se le denomina canal. En este sentido, la información de las imágenes es un tensor con 3 canales, ya que, posee píxeles a lo largo de dos dimensiones espaciales (ancho y largo) y, en cada canal se almacena la información del color rojo, verde y azul, respectivamente.

En el transcurso de la información por la red convolucional se transforman las dimensiones originales de la información. Por un lado, en las capas convolucionales, se crean nuevos canales asociados a cada Kernel (lo que modifica la tercera dimensión del tensor), y se puede comprimir la información (i.e. disminuir las dos primeras dimensiones del tensor) aunque es poco habitual. Por otro lado, la capa de pooling también posee la capacidad de comprimir la información de entrada, siendo común la compresión de la información en esta capa.

A lo largo de la red convolucional, la información se comprime cada vez más y se generan más canales. Por lo tanto, la red convolucional inicialmente identifica patrones sencillos (información sin comprimir) y escasos (bajo número de canales) y, en las últimas capas se maneja patrones complejos (información comprimida) y numerosos (elevado número de canales), como se muestra en la figura 3.16.

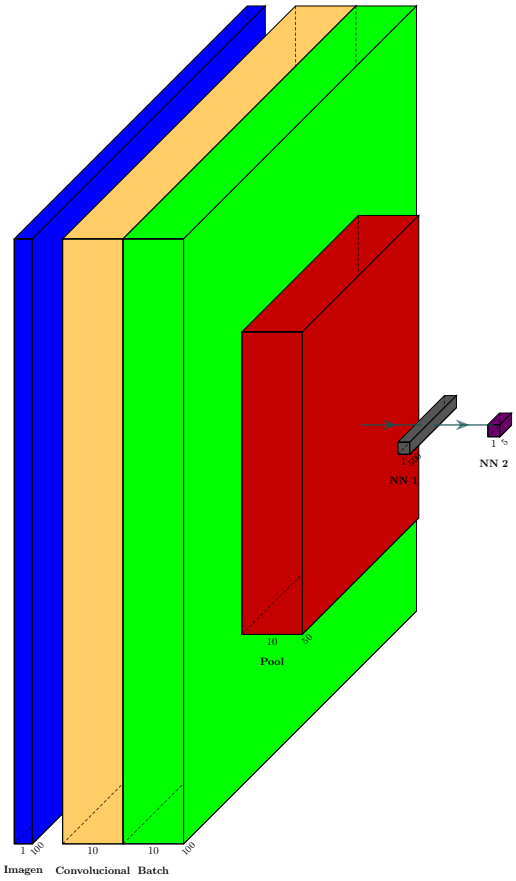


Figura 3.16: Ejemplo de red neuronal convolucional. La profundidad, altura y anchura de los prismas representan la anchura, altura y número de canales, respectivamente. El prisma azul, crema, verde, rojo, gris y morado representan la imagen original, capa convolucional, capa de normalización, capa de pooling, capa neuronal de entrada y capa neuronal de salida, respectivamente.

Capítulo 4

Base de datos

4.1 Base de datos original

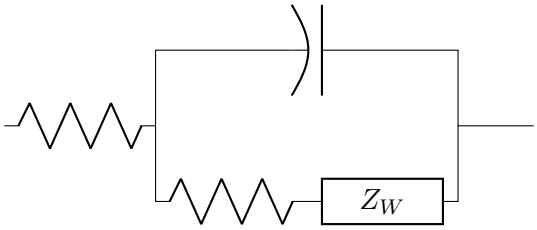
La base de datos original utilizada en este Trabajo Final de Máster (TFM) es una recopilación de datos de espectros EIS para 5 tipos de circuitos eléctricos equivalentes (mostrados en la figura 4.1) publicados en artículos científicos. Esta ha sido desarrollada por el grupo de Shan Zu y presentada en su artículo "Equivalent circuit model recognition of electrochemical impedance spectroscopy via machine learning" (Zhu et al. 2019) que contiene un link para acceder de forma libre a la base de datos a través de la plataforma Github (`https://github.com/Shan-Zhu/ML-EIS/blob/master/EIS-Data-Raw.csv`).

La construcción de la base de datos se explica en el artículo de Zhu (Zhu et al. 2019). Básicamente, seleccionaron una extensa bibliografía científica de artículos que presentaban espectros EIS experimentales, en los que los autores asociaban el espectro a uno de los 5 circuitos eléctricos equivalentes considerados. Tras ello, se digitalizaron los diagramas de Nyquist con la herramienta online WebPlotDigitizer, que permite leer gráficas de forma precisa. Con WebPlotDigitizer leyeron 80 puntos por espectro EIS, estando cada punto formado por dos valores (valor del eje real y eje imaginario). Por lo tanto, se tiene un total de 160 valores para describir cada espectro EIS (los 80 primeros valores del eje real y, los 80 restantes del eje imaginario).

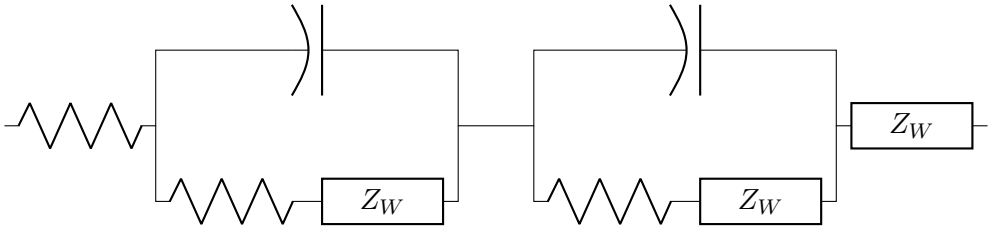
La base de datos está organiza de la siguiente manera: La primera fila contiene un identificador del espectro. La segunda fila contiene el DOI del artículo de origen. La tercera fila contiene el tipo de circuito eléctrico equivalente. El resto de filas (i.e. las 160 restantes) guardan información sobre el espectro EIS correspondiente. En cuanto a las columnas guardan la información comentada para cada uno de los 502 espectros que contiene, como se muestra en la figura 4.2.

La base de datos presenta diversos problemas conceptuales:

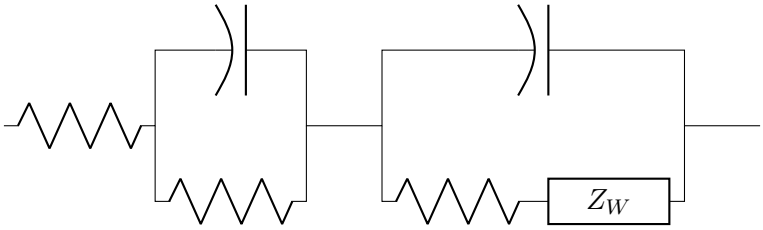
- Los diagramas de Nyquist publicados en la mayoría de artículos científicos no etiquetan la frecuencia y, por lo tanto, la base de datos describe espectros EIS de forma incompleta (i.e. carece de datos de frecuencia).



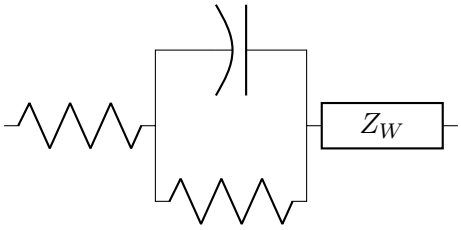
(a) Circuito eléctrico equivalente tipo 1



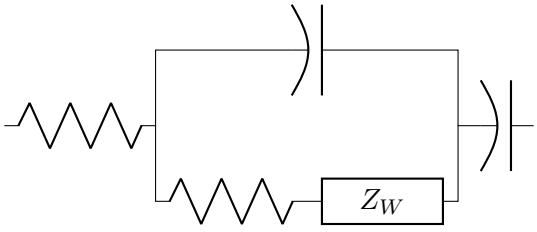
(b) Circuito eléctrico equivalente tipo 2



(c) Circuito eléctrico equivalente tipo 3



(d) Circuito eléctrico equivalente tipo 4



(e) Circuito eléctrico equivalente tipo 5

Figura 4.1: Circuitos eléctricos equivalentes considerados en la base de datos. Adaptado de Zhu et al. 2019.

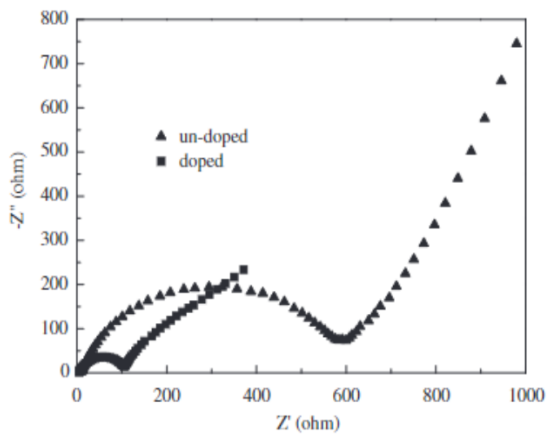
Column1	Column2	Column3	Column4	Column5
DOI	10.1016/j.ceramint.2013.10.055	10.1016/j.ceramint.2013.10.055	10.1007/s10570-014-0274-7	10.1007/s10570-014-0274-7
Type	type-1	type-1	type-1	type-1
Z'(1)	20.6	30.42	15.9	12.42
Z'(2)	28.85	46.53	15.91	14.17
Z'(3)	28.85	61.56	19.43	15.92
Z'(4)	28.89	78.63	22.95	17.7
Z'(5)	37.13	96.69	22.98	17.71
Z'(6)	37.15	115.81	25.18	19.43
Z'(7)	37.16	134.94	28.79	22.33
Z'(8)	53.73	154.06	34	23.83
Z'(9)	74.46	173.19	38.02	25.68
Z'(10)	78.59	194.71	43.61	28.2
Z'(11)	78.64	218.63	44.28	28.32
Z'(12)	78.68	242.56	49.24	30.19
Z'(13)	82.73	266.49	49.27	31.54
Z'(14)	95.2	290.42	54.6	35.55
Z'(15)	95.22	314.35	58.46	36.73
Z'(16)	107.65	338.28	64.22	41.16
Z'(17)	120.09	362.21	66.77	43.42
Z'(18)	125.42	386.16	73	50.79
Z'(19)	142.16	410.11	85.35	52.37
Z'(20)	158.91	434.06	94.01	58.79

Figura 4.2: Ejemplo ilustrativo de la estructura de los datos en la base de datos original.

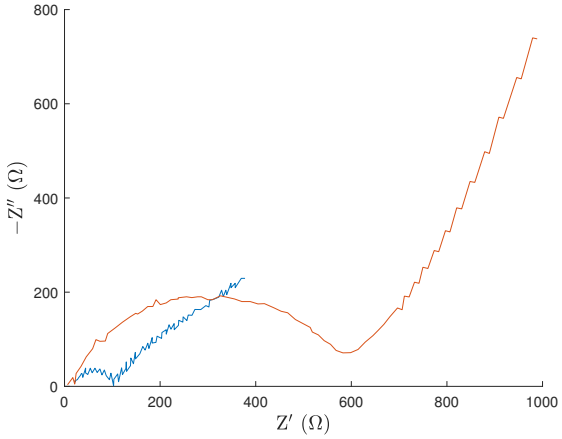
- La clasificación realizada por los autores de los artículos científicos de base puede no ser adecuada, lo que unido a la capacidad de un espectro EIS de ser compatible con varios circuitos equivalentes (como se muestra en la figura 4.5), puede confundir a la inteligencia artificial.

Por otro lado, la base de datos presenta problemas de ejecución:

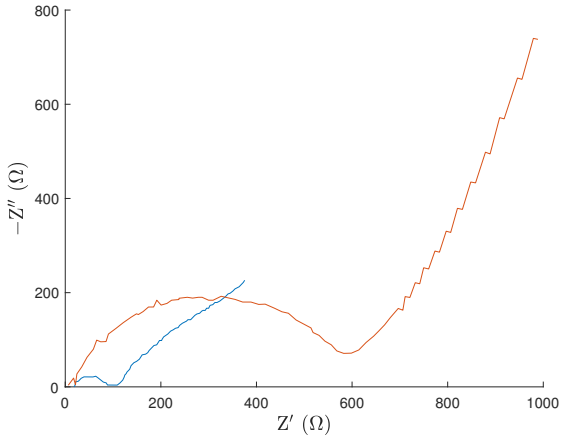
- Ruido al leer los diagramas con WebPlotDigitizer, como se muestra en la figura 4.3b.
- Espectros EIS clasificados de forma diferente en la base de datos y el artículo de referencia
- Puntos leídos de forma desordenada, lo que genera problemas de representación, como se muestra en la figura 4.4b.
- Es una base de datos descompensada: Los circuitos eléctricos equivalentes considerados no tienen el mismo número de observaciones (i.e. la clase 1, 2, 3, 4 y 5 representan el 38.84 %, 19.72 %, 16.53 %, 15.73 % y 9.16 %, respectivamente). Esto implica que a la hora de entrenar a la inteligencia artificial se tienen más observaciones para algunas clases y, por lo tanto, la clasificación realizada por la inteligencia artificial estará condicionada por esta asimetría en la base de datos.
- Los 80 valores de cada espectro son seleccionados sin tener en cuenta a que frecuencia corresponden y, por lo tanto, los valores de cada espectro no son comparables.



(a) Espectro EIS original.

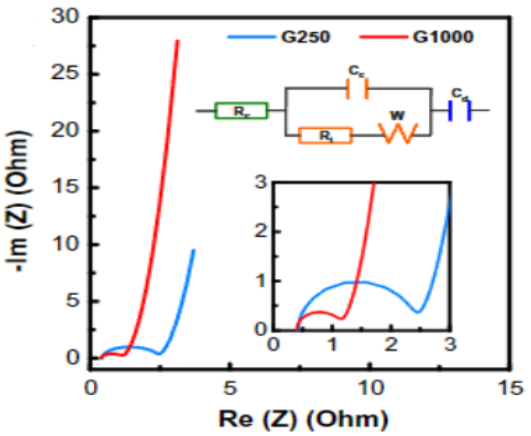


(b) Espectro EIS en la base de datos original.

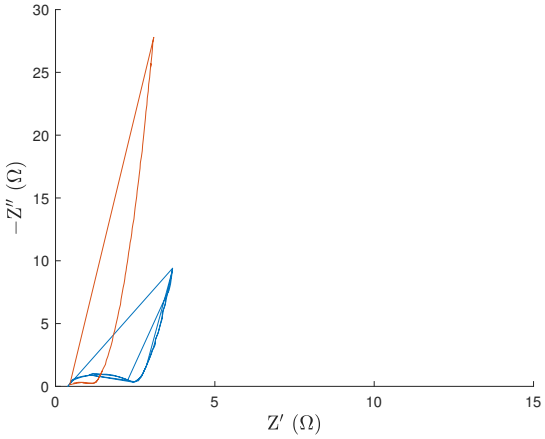


(c) Espectro EIS en la base datos refinada.

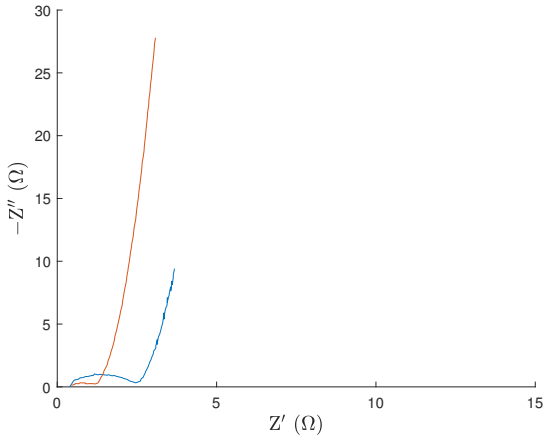
Figura 4.3: Espectro EIS número 14. (a) diagrama de Nyquist del artículo original (sin ruido), (b) diagrma de Nyquist de la base de datos original (con ruido) y (c) diagrama Nyquist con ruido en la base de datos refinada (sin ruido).



(a) Espectro EIS original.

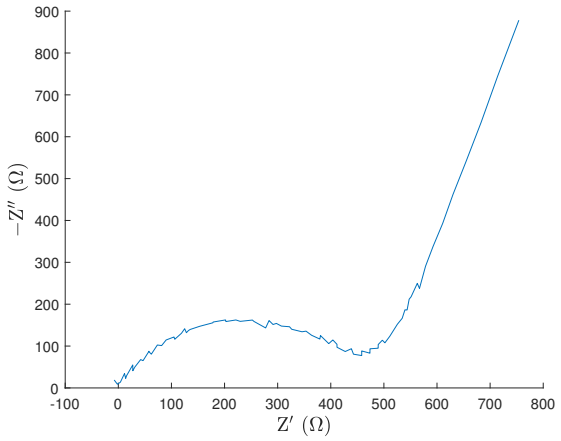


(b) Espectro en la base de datos original.

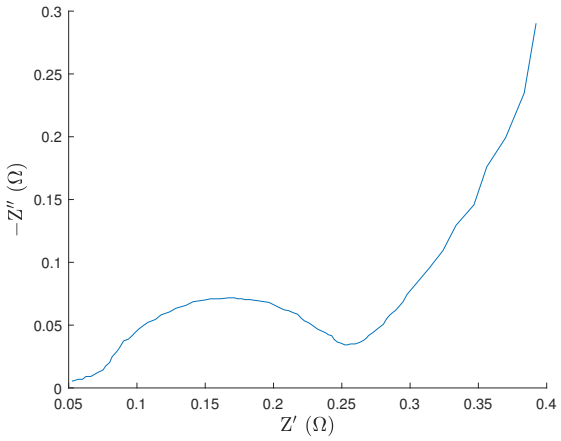


(c) Espectro EIS en la base de datos refinada.

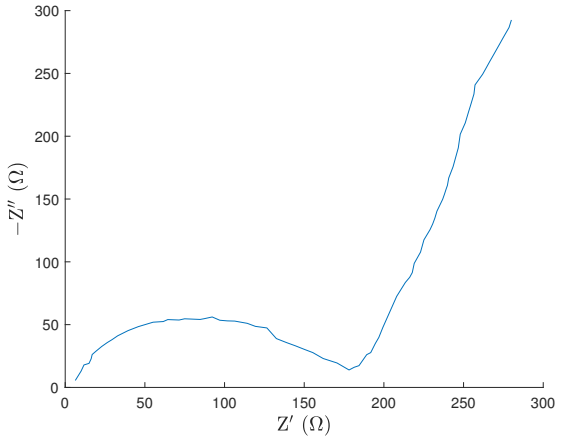
Figura 4.4: Espectros EIS 501 y 502 de la base de datos refinada. (a) diagrama de Nyquist del artículo original (ordenado), (b) diagrama de Nyquist de la base de datos original (desordenado) y (c) diagrama de Nyquist de la base de datos refinada (ordenado).



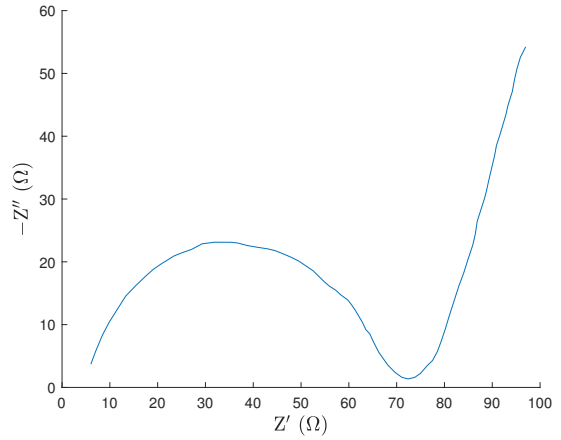
(a) Espectro 123 perteneciente a la clase 1



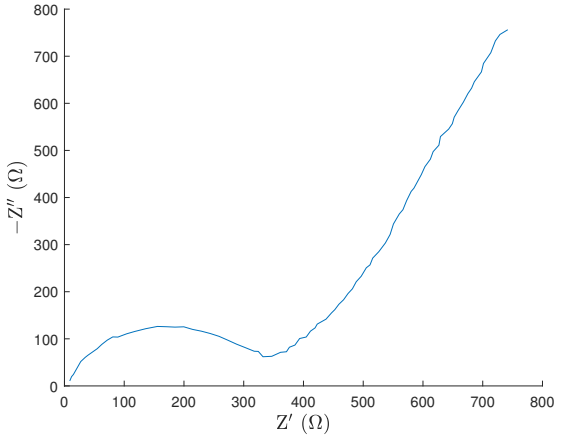
(b) Espectro 206 perteneciente a la clase 2



(c) Espectro 346 perteneciente a la clase 3



(d) Espectro 379 perteneciente a la clase 4



(e) Espectro 479 perteneciente a la clase 5

Figura 4.5: Circuitos eléctricos equivalentes de la base de datos original (sin ruido y ordenados) pertenecientes a las clases 1 (a), 2 (b), 3 (c), 4 (d) y 5 (e) con formas similares.

4.2 Base de datos refinada

Puesto que, una buena base de datos es condición necesaria para obtener un buen algoritmo de recomendación, en este Trabajo Final de Máster (TFM) se ha creado una base de datos refinada (a partir de la base de datos descrita en el apartado 4.1), cuya estructura es la siguiente: La primera fila, contiene el DOI del artículo. La segunda fila, contiene el estado del espectro en la base de datos original. La tercera, contiene el identificador del espectro en la base de datos original. La cuarta fila, contiene un identificador visual del espectro (i.e. color, forma geométrica), lo que permite distinguir diferentes espectros de un mismo artículo. La quinta fila, contiene el circuito eléctrico equivalente correspondiente. La sexta fila, contiene el identificador de tipo de circuito eléctrico equivalente según el artículo al que pertenece. El resto de filas (i.e. las 160 restantes) guardan información sobre el espectro EIS correspondiente. En cuanto a las columnas guardan la información comentada para cada uno de los 544 espectros que contiene, como se muestra en la figura 4.6.

Los cambios realizados en la base de datos refinada son:

- Los espectros EIS con ruido han sido sustituidos por espectros EIS sin ruido. Estos han sido generados con los diagramas de Nyquist de la referencia original y, leídos correctamente con la herramienta WebPlotDigitizer, como se muestra en la figura 4.3c.
- Los espectros EIS cuya clase en la base de datos original y su referencia no coincidía, se han clasificado correctamente.
- Los espectros EIS desordenados han sido ordenados mediante un algoritmo de Matlab®, como se muestra en la figura 4.4c.
- Introducción de nuevos espectros EIS, pasando de 502 a 544. Con esta expansión se ha intentado aumentar preferentemente el número de espectros de las clases menos pobladas, siendo los nuevos porcentajes de la clase 1, 2, 3, 4 y 5, 34.37 %, 22.24 %, 17.64 %, 15.80 % y 9.92 %, respectivamente.

Referencia	10.1016/j.ceramint.2013.10.055	10.1016/j.ceramint.2013.10.055	10.1016/j.ceramint.2013.10.055	10.1016/j.ceramint.2013.10.055
Estado Original	Mal	Ok	Ok	Ok
Nº Espectro Antiguo	7	8	1	2
Color/Ident	Verde	Morado	Rojo	Azul
Nº Espectro	1	2	3	4
Tipo	1	1	1	1
	5.22318272	8.27	20.6	30.42
	36.60789636	8.29	28.85	46.53
	73.03332345	12.38	28.85	61.56
	117.1452683	20.64	28.89	78.63
	174.0528705	28.95	37.13	96.69
	207.6956413	28.97	37.15	115.81
	230.9604726	45.5	37.16	134.94
	256.7798736	45.56	53.73	154.06
	282.6220833	45.59	74.46	173.19
	303.195493	65.72	78.59	194.71
	326.3690897	82.82	78.64	218.63
	357.2292042	103.21	78.68	242.56
	375.1796182	120.76	82.73	266.49
	390.5982711	139.92	95.2	290.42
	403.416737	159.07	95.22	314.35
	418.8581986	178.24	107.65	338.28
	429.1220947	197.41	120.09	362.21
	439.3631823	216.58	125.42	386.16

Figura 4.6: Ejemplo ilustrativo de la estructura de los datos en la base de datos refinada.

Capítulo 5

Implementación y caracterización de los algoritmos

5.1 Implementación

5.1.1 Algoritmos de Machine Learning

Support Vector Machine

El algoritmo de Support Vector Machine (SVM) se implementó mediante la función nativa de Matlab® *fitcsvm*. Esta permite la elección de diferentes Kernels (lineal, cuadrático, cúbico, etc), regular la aversión a cometer errores mediante el Box Constraint, escalar los datos de entrada mediante el Kernel Scale, y minimizar la función de coste asociada a este algoritmo (mostrada en el capítulo 3).

Neural Network

El algoritmo de Neural Network (NN) se implementó mediante la función nativa de Matlab® *fitcnet*. Esta permite la customización de todos los hiperparámetros de una red neuronal (aunque, en este caso se elige la red neuronal por defecto, modificando únicamente el número de neuronas y la función de activación para la capa oculta) y minimización de la función de coste.

El proceso de entrenamiento del algoritmo NN consiste en el ajuste de la matriz de pesos ($\mathbf{W}$) y el vector de sesgos ($\vec{b}$). Para ello, se establece una función de coste dependiente de $\mathbf{W}$ y $\vec{b}$, que en el caso de la función fitcnet de Matlab® es el entrecruzamiento entrópico:

$$EE = -\sum_{i=1}^{n} y_i \cdot \mathbb{P}(y_i) \tag{5.1}$$

Siendo EE el valor del entrecruzamiento entrópico, n el número de clases y $\mathbb{P}(y_i)$ la probabilidad estimada para la clase y_i por la red neuronal.

Dicha función de coste es minimizada con el algoritmo limited-memory Broyden-Fletcher-Goldfarb-Shanno quasi-Newton algorithm (LBFGS) que, aproxima la matriz Hessiana (por ello, el nombre de quasi-Newton) a partir de los gradientes más recientes, lo que limita su uso de memoria (por ello, el nombre de limited-memory).

Naive Bayes

El algoritmo de Naive Bayes (NB) se implementó mediante la función nativa de Matlab® *fitcnb*, que permite la elección de diferentes distribuciones (e.g. normal, box, etc). Sin embargo, no minimiza ninguna función de coste, ya que, para este caso no existe. En cambio, el proceso de entrenamiento del algoritmo NB se basa en la estimación de probabilidades del set de entrenamiento, como se detalla en el capítulo 3.

K Nearest Neighbor

El algoritmo K Nearest Neighbor (KNN) se implementó mediante la función nativa de Matlab® *fitcknn*, que permite la elección de diferentes hiperparámetros (número de vecinos y medida de la distancia). Sin embargo, no minimiza ninguna función de coste, ya que, para este caso no existe. En cambio, el proceso de entrenamiento del algoritmo NB se basa en el guardado los datos de entrenamiento, como se detalla en el capítulo 3.

Decision Tree

El algoritmo Decision Tree (DT) se implementó mediante la función nativa de Matlab® *fitctree*. Esta permite la elección del número de divisiones máximo y minimizar la función de coste.

El entrenamiento del algoritmo DT consiste en maximizar la homogeneidad de los nodos, lo que puede realizarse con funciones de coste que cuantifiquen el desorden, como por ejemplo el Índice de Gini:

$$IG = 1 - \sum_{i=1}^{n} \mathbb{P}_i^2 \tag{5.2}$$

Siendo IG el valor del Índice de Gini y $\mathbb{P}_i$ la probabilidad que el elemento i-ésimo sea clasificado incorrectamente.

5.1.2 Algoritmos de Deep Learning

Convolutional Neural Network

El algoritmo red convolucional o CNN por sus siglas en inglés (Convolutional Neural Network) se implementó mediante la función nativa de Matlab® *trainNetwork*. Esta permite la modificación de todos los hiperparámetros asociados con el algoritmo CNN (i.e. número de capas convolucionales, Stride, Padding, etc) y la minimización de la función de coste seleccionada.

El proceso de entrenamiento de las redes convolucionales está basado en la obtención de los coeficientes para los Kernels, capas de normalización y, la obtención de pesos y sesgos para las capas de redes neuronales. Esto se realiza con el optimizador de descenso de gradiente estocástico con momento o SGDM por sus siglas en inglés (Stochastic Gradient Descent with Momentum), que utiliza la siguiente expresión para actualizar los parámetros optimizables:

$$\psi_{i+1} = \psi_i - \text{ILR} \cdot \nabla f_{cost}(\psi_i) + \gamma \cdot (\psi_i - \psi_{i-1}) \tag{5.3}$$

Donde ψ es el vector de parámetros optimizables, ILR es el Initial Learn Rate, f_{cost} es la función de coste, y γ el momento.

Este algoritmo no calcula el gradiente de la función de coste de forma exacta (por ello, denominado gradiente estocástico), ya que, suele implicar un elevado número de cálculos. En cambio, selecciona un subconjunto de ejemplos de entrenamiento (cuyo número en Matlab® es denominado MiniBatchSize) y, estima el gradiente con esa aproximación.

La contribución del gradiente estocástico está regulada por el parámetro ILR, que permite modificar la velocidad de convergencia. Por un lado, valores elevados de ILR ralentizan la convergencia, pero disminuyen la búsqueda errática de soluciones. Por otro lado, valores pequeños aumentan la velocidad de la convergencia, pero propician la búsqueda errática de soluciones.

Además, el algoritmo SGDM introduce un término adicional que disminuye la oscilación en la búsqueda de soluciones. Este término está ligado al cambio en los parámetros optimizables de la iteración anterior (i.e. $\psi_i - \psi_{i-1}$) y, está regulado por el momento (γ).

Este proceso de entrenamiento debe ser aplicado a una arquitectura de red convolucional concreta. Por ello, se implementaron diferentes arquitecturas de redes convolucionales, como se explica a continuación:

Para la generación de las arquitecturas convolucionales consideradas, se comenzó con 3 arquitecturas base que aceptan una imagen de resolución 100x100 píxeles y, difieren en la rapidez de compresión de la información en la parte convolucional, como se muestra en la figura 5.1.

La primera arquitectura (mostrada en la figura 5.1a) está compuesta por 3 etapas convolucionales (i.e. la concatenación de una capa convolucional, normalización y pool) y 1 etapa densa (i.e. 2 capas de redes neuronales, una para captar la información de las etapas convolucionales y otra para transmitir la información al exterior). La principal característica de esta arquitectura es que no comprime la información a través de las etapas convolucionales, pero si que aumenta el número de canales. Esta información y, el resto de características asociadas a la arquitectura de la red convolucional 1 se recogen en la tabla 5.1.

La segunda arquitectura (mostrada en la figura 5.1b) está compuesta por 6 etapas convolucionales y 1 etapa densa. La primera etapa convolucional genera 10 canales y, el resto incrementa en 5 canales los canales de la etapa convolucional anterior (i.e la etapa 2 tiene 15 canales, la etapa 3 20, etc). Sin embargo, la compresión de la información se realiza únicamente en las etapas pares. La etapa densa tiene la misma estructura que en la arquitectura 1 pero, con menos neuronas en la primera capa neuronal puesto que se ha comprimido la información en las etapas convolucionales (i.e. en la etapa convolucional 2, 4 y 6). Esta información y, el resto de características asociadas a la arquitectura de la red convolucional 2 se recogen en la tabla 5.2.

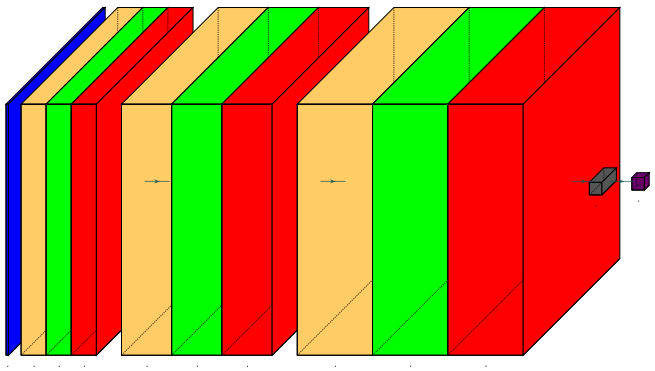
La tercera arquitectura (mostrada en la figura 5.1c) está compuesta por 3 etapas convolucionales y 1 densa. La primera etapa convolucinal genera 10 canales y, el resto incrementa en 10 canales los canales de la etapa convolucional anterior (i.e la etapa 2 tiene 20 canales y la etapa 3 30 canales). La compresión de información es la más brusca de las 3 arquitecturas, ya que, cada capa convolucional comprime la información. Esta información y, el resto de características asociadas a la arquitectura de la red convolucional 3 se recogen en la tabla 5.4.

A partir de las 3 arquitecturas base de redes convolucionales, se generaron 9 arquitecturas diferentes. Para ello, se reemplazaron las funciones de activación ReLU por las funciones de activación tangente hiperbólica y Leaky ReLU. Por lo tanto, de cada arquitectura base se obtuvieron 3 arquitecturas en total (la arquitectura base y otras dos).

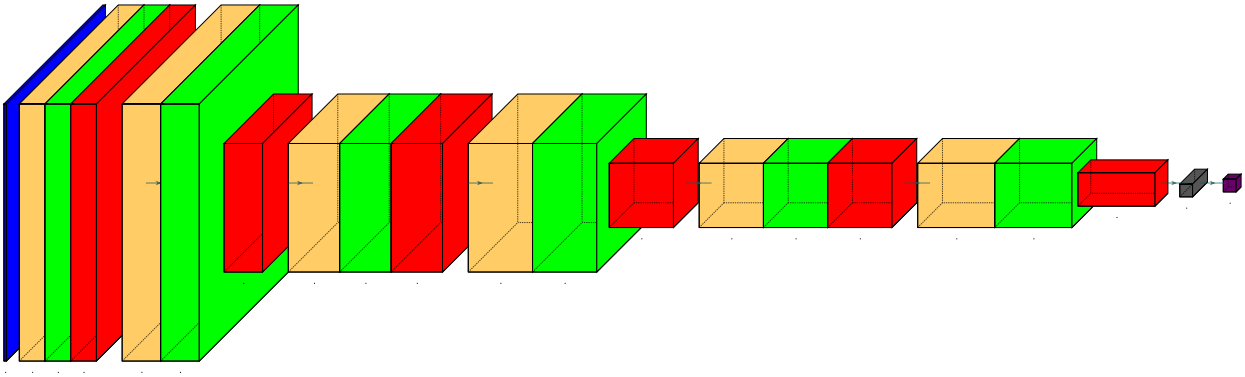
La denominación que recibe cada una de ellas está compuesta por dos números separados por un punto, donde el primero hace referencia a la arquitectura base y el segundo a la función de activación. Por ejemplo, para la arquitectura base 1, se obtienen las variantes 1.1, 1.2 y 1.3, para las funciones de activación ReLU, tangente hiperbólica y Leaky ReLU, respectivamente.

A partir de las redes convolucionales comentadas, se pueden generar más redes convolucionales mediante la modificación de la parte densa. En este caso, la modificación consistió en la introducción de una capa neural extra con 10 neuronas, ratio de aprendizaje 1 y la misma función de activación que el resto de la red convolucional.

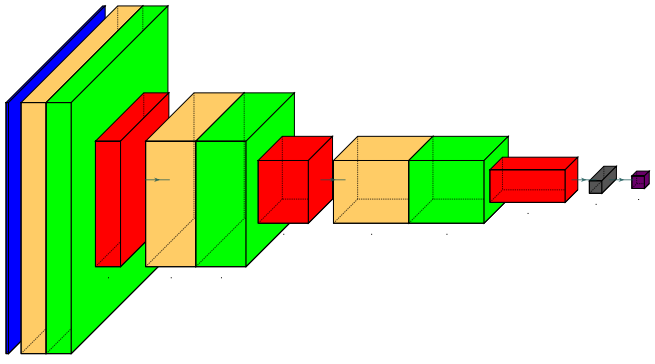
Por lo tanto, para cada arquitectura mostrada en el apartado anterior se generaron 2, una correspondiente a la original y la otra con la parte densa modificada. La denominación para referirse a estas arquitecturas será añadir un número más a la nomenclatura, denotando 1 y 2 la ausencia y presencia de la capa neuronal extra, respectivamente. Por ejemplo, de la arquitectura 1.1 se generaron 2 arquitecturas 1.1.1 y 1.1.2 para la arquitectura sin la capa neuronal extra y la arquitectura con la capa neuronal extra, respectivamente.



(a) Arquitectura 1



(b) Arquitectura 2



(c) Arquitectura 3

Figura 5.1: Imágenes ilustrativas de las 3 arquitecturas base de redes convolucionales consideradas. La capa azul hace referencia a la imagen de entrada, las capas color crema, verde y rojo hacen referencia a las capas covolucionales, normalización y pool, respectivamente. Finalmente, las capas de color gris y morado hacen referencia a la parte densa.

Tabla 5.1: Información de la arquitectura de red convolucional 1.

Etapa	Capa	Características	Valor
Convolucional 1	Convolucional	Estructura datos de salida	100x100x10
		Número de filtros	10
		Función de activación	ReLU
		Tamaño de filtro	3x3
		Stride	1x1
		Padding	Same
		Ratio de aprendizaje	1
	Batch normalization	Estructura datos de salida	100x100x10
		Ratio de aprendizaje	1
	Average pooling	Estructura datos de salida	100x100x10
		Pool size	2x2
		Stride	1x1
		Padding	Same
Convolucional 2	Convolucional	Estructura datos de salida	100x100x20
		Número de filtros	20
		Función de activación	ReLU
		Tamaño de filtro	3x3
		Stride	1x1
		Padding	Same
		Ratio de aprendizaje	1
	Batch normalization	Activaciones	100x100x20
		Ratio de aprendizaje	1
	Average pooling	Estructura datos de salida	100x100x20
		Pool size	2x2
		Stride	1x1
		Padding	Same
Convolucional 3	Convolucional	Estructura datos de salida	100x100x30
		Número de filtros	30
		Función de activación	ReLU
		Tamaño de filtro	3x3
		Stride	1x1
		Padding	Same
		Ratio de aprendizaje	1
	Batch normalization	Estructura datos de salida	100x100x30
		Ratio de aprendizaje	1
	Average pooling	Estructura datos de salida	100x100x30
		Pool size	2x2
		Stride	1x1
		Padding	Same
Densa	Capa densa 1	Estructura datos de salida	300 000x1x1
		Número neuronas	300 000
	Densa 2	Estructura datos de salida	5x1x1
		Número neuronas	5
		Función de activación	Softmax
		Ratio aprendizaje	1

Tabla 5.2: Arquitectura de la red convolucional 2.

Etapa	Capa	Características	Valor
Convolucional 1	Convolucional	Estructura datos de salida	100x100x10
		Número de filtros	10
		Función de activación	ReLU
		Tamaño de filtro	3x3
		Stride	1x1
		Padding	Same
		Ratio de aprendizaje	1
	Batch normalization	Estructura datos de salida	100x100x10
		Ratio de aprendizaje	1
	Average pooling	Estructura datos de salida	100x100x10
		Pool size	2x2
		Stride	1x1
		Padding	Same
Convolucional 2	Convolucional	Estructura datos de salida	100x100x15
		Número de filtros	15
		Función de activación	ReLU
		Tamaño de filtro	5x5
		Stride	1x1
		Padding	Same
		Ratio de aprendizaje	1
	Batch normalization	Activaciones	100x100x15
		Ratio de aprendizaje	1
	Average pooling	Estructura datos de salida	50x50x15
		Pool size	2x2
		Stride	2x2
		Padding	Same
Convolucional 3	Convolucional	Estructura datos de salida	50x50x20
		Número de filtros	20
		Función de activación	ReLU
		Tamaño de filtro	5x5
		Stride	1x1
		Padding	Same
		Ratio de aprendizaje	1
	Batch normalization	Estructura datos de salida	50x50x20
		Ratio de aprendizaje	1
	Average pooling	Estructura datos de salida	50x50x20
		Pool size	2x2
		Stride	1x1
		Padding	Same

Tabla 5.3: Continuación de la tabla 5.2.

Etapa	Capa	Características	Valor
Convolucional 4	Convolucional	Estructura datos de salida	50x50x25
		Número de filtros	25
		Función de activación	ReLU
		Tamaño de filtro	5x5
		Stride	1x1
		Padding	Same
		Ratio de aprendizaje	1
	Batch normalization	Estructura datos de salida	50x50x25
		Ratio de aprendizaje	1
	Average pooling	Estructura datos de salida	25x25x25
		Pool size	2x2
		Stride	2x2
		Padding	Same
Convolucional 5	Convolucional	Estructura datos de salida	25x25x25
		Número de filtros	25
		Función de activación	ReLU
		Tamaño de filtro	5x5
		Stride	1x1
		Padding	Same
		Ratio de aprendizaje	1
	Batch normalization	Estructura datos de salida	25x25x25
		Ratio de aprendizaje	1
	Average pooling	Estructura datos de salida	25x25x25
		Pool size	2x2
		Stride	1x1
		Padding	Same
Convolucional 6	Convolucional	Estructura datos de salida	25x25x30
		Número de filtros	30
		Función de activación	ReLU
		Tamaño de filtro	7x7
		Stride	1x1
		Padding	Same
		Ratio de aprendizaje	1
	Batch normalization	Estructura datos de salida	25x25x30
		Ratio de aprendizaje	1
	Average pooling	Estructura datos de salida	13x13x30
		Pool size	2x2
		Stride	2x2
		Padding	Same
Neuronal densa	Densa 1	Estructura datos de salida	5070x1x1
		Número neuronas	5070
	Densa 2	Estructura datos de salida	5x1x1
		Número neuronas	5
		Función de activación	Softmax
		Ratio aprendizaje	1

Tabla 5.4: Parámetros seleccionados en la arquitectura de red convolucional 3

Etapa	Capa	Características	Valor
Convolucional 1	Convolucional	Estructura datos de salida	100x100x10
		Número de filtros	10
		Función de activación	ReLU
		Tamaño de filtro	3x3
		Stride	1x1
		Padding	Same
		Ratio de aprendizaje	1
	Batch normalization	Estructura datos de salida	100x100x10
		Ratio de aprendizaje	1
	Average pooling	Estructura datos de salida	50x50x10
		Pool size	2x2
		Stride	2x2
		Padding	Same
Convolucional 2	Convolucional	Estructura datos de salida	50x50x20
		Número de filtros	20
		Función de activación	ReLU
		Tamaño de filtro	5x5
		Stride	1x1
		Padding	Same
		Ratio de aprendizaje	1
	Batch normalization	Estructura datos de salida	50x50x20
		Ratio de aprendizaje	1
	Average pooling	Estructura datos de salida	25x25x20
		Pool size	2x2
		Stride	2x2
		Padding	Same
Convolucional 3	Convolucional	Estructura datos de salida	25x25x30
		Número de filtros	30
		Función de activación	ReLU
		Tamaño de filtro	5x5
		Stride	1x1
		Padding	Same
		Ratio de aprendizaje	1
	Batch normalization	Estructura datos de salida	25x25x30
		Ratio de aprendizaje	1
	Average pooling	Estructura datos de salida	13x13x30
		Pool size	2x2
		Stride	2x2
		Padding	Same
Neuronal densa	Densa 1	Estructura datos de salida	5070x1x1
		Número neuronas	5070
	Densa 2	Estructura datos de salida	5x1x1
		Número neuronas	5
		Función de activación	Softmax
		Ratio aprendizaje	1

5.2 Metodología de caracterización

5.2.1 Definiciones básicas

Exactitud

La exactitud es la magnitud utilizada para evaluar el desempeño de los clasificadores. Esta puede definirse como el porcentaje de predicciones correctas entre todas la predicciones realizadas por un clasificador, lo que matemáticamente puede expresarse:

$$Exactitud = \frac{1}{n} \cdot \left(\sum_{i=1}^{n} (\hat{y}_i = y_i) \right) \cdot 100 \tag{5.4}$$

Siendo n el número de predicciones totales, y_i la clase verdadera del ejemplo i-ésimo, $\hat{y}_i$ la predicción del clasificador para el ejemplo i-ésimo, y $\hat{y}_i = y_i$ una función lógica, que vale 1 si la predicción y la clase coinciden, siendo 0 en otro caso.

5.2.2 Optimización de hiperparámetros

El estudio de la exactitud para los algoritmos considerados, se realizó del siguiente modo:

En primer lugar, se dividieron los espectros contenidos en la base de datos refinada en los sets de entrenamiento (80 % de los espectros) y test (20 % de los espectros) de forma aleatoria. En segundo lugar, se utilizó el set de entrenamiento para generar un clasificador basado en un algoritmo de Machine Learning con una combinación específica de hiperparámetros. En tercer lugar, el clasificador se utilizó para generar predicciones del set de entrenamiento y test. En cuarto lugar, se utilizaron los datos obtenidos en el tercer paso para calcular la exactitud conforme lo expuesto en la sección 5.2.1. En quinto lugar, se repitieron los pasos segundo, tercero y cuarto cambiado la elección de hiperparámetros (i.e. se repitió el proceso de generación del clasificador y cálculo de exactitud para diferentes combinaciones de hiperparámetros). En sexto lugar, se repitieron todos los pasos anteriores un número determinado de veces (i.e. se repite el proceso de generación de clasificadores y medida de exactitud para diferentes hiperparámetros y sets de entrenamiento y test).

Los detalles de esta metodología aplicada a cada uno de los algoritmos es la siguiente:

- Support Vector Machine: La optimización de hiperparámetros para diferentes combinaciones de hiperparámetros (i.e. Box Constraint, Kernel Scale y Kernel) se realizó con la función *fitcsvm* mediante el procedimiento explicado anteriormente. Este se repitió 50 veces para cada combinación de hiperparámetros (con sets de entrenamiento y test diferentes), excepto las combinaciones del Kernel linear que se repitieron 10 veces (con sets de entrenamiento y test diferentes).
- Neural Network: La optimización de hiperparámetros para diferentes combinaciones de hiperparámetros (i.e. diferentes combinaciones de funciones de activación y número de neuronas en la capa oculta) se realizó con la función *fitcnet* mediante el procedimiento explicado anteriormente. Este se repitió para cada combinación de hiperparámetros 2000 veces (con sets de entrenamiento y test diferentes).

- Naive Bayes: La optimización del hiperparámetro distribución (normal, box, epanechnikov y triangle) se realizó con la función *fitcnb* mediante el procedimiento descrito anteriormente. Este se repitió 2000 veces (con sets de entrenamiento y test diferentes) para cada valor del hiperparámetro (con sets de entrenamiento y test diferentes).
- K Nearest Neighbor: La optimización de hiperparámetros para diferentes combinaciones de hiperparámetros (i.e. diferentes combinaciones de métricas de distancia y número de vecinos) se realizó con la función *fitcknn* mediante el procedimiento descrito anteriormente. Este se repitió 2000 veces (con sets de entrenamiento y test diferentes) para cada combinación de hiperparámetros (con sets de entrenamiento y test diferentes).
- Decision Tree: La optimización del hiperparámetro número de divisiones se realizó con la función *fitctree* mediante el procedimiento descrito anteriormente. Este se repitió 2000 veces para cada valor del hiperparámetro (con sets de entrenamiento y test diferentes).

Sin embargo, el algoritmo Convolutional Neural Network debido a limitaciones de computacionales, requirió una metodología diferente:

En primer lugar, se generó un base de datos estática (i.e. una base de datos cuyos sets de test y entrenamiento no varían), cuyo set de test estaba formado por los primeros espectros de cada clase hasta alcanzar el 20 % y el resto de espectros formaba parte del set de entrenamiento. Esto permitió eliminar la variabilidad en la exactitud obtenida por la elección aleatoria de los sets de entrenamiento y test. En segundo lugar, se utilizó el set de entrenamiento del paso anterior para entrenar una única vez a las 27 redes convolucionales detalladas en la sección 5.1.2. En tercer lugar, se utilizaron los datos obtenidos en el segundo paso para obtener la exactitud según lo expuesto en la sección 5.2.1. En cuarto lugar, se seleccionaron las 3 arquitecturas convolucionales con mayor exactitud del paso anterior, se congeló su parte convolucional (i.e. se bloquearon los parámetros optimizables de las capas convolucionales) y, se modificó su parte densa según lo expuesto en la sección 5.1.2. Con ello se obtuvieron 6 arquitecturas convolucionales (3 con la parte densa no modificada y 3 con la parte densa modificada) con la parte convolucional congelada. En quinto lugar, se dividieron los espectros contenidos en la base de datos refinada en los sets de entrenamiento y test de forma aleatoria. En sexto lugar, se utilizó el set de entrenamiento del quinto paso para entrenar a las redes convolucionales obtenidas en el cuarto paso. En séptimo lugar, se utilizaron los datos obtenidos en el sexto paso para obtener la exactitud según lo expuesto en la sección 5.2.1. En octavo lugar, se repitieron los pasos quinto, sexto y séptimo 2000 veces (i.e. se repitió 2000 veces el entrenamiento de las mejores redes convolucionales para diferentes sets de entrenamiento y test).

5.2.3 Matrices de confusión

En este Trabajo Final de Máster (TFM) se utilizaron dos matrices de confusión diferentes, la matriz de confusión desagregada y agregada. La primera, muestra en el eje vertical el número de espectro, en el eje horizontal la clase predicha y, a través de una escala de color indica el número de veces que un espectro ha sido clasificado como una clase en concreto (en tanto por 1).

La segunda, muestra en el eje vertical la clase verdadera, en el eje horizontal la clase predicha por el clasificador y, a través de una escala de color indica el número de veces que una clase ha sido clasificada como una clase en concreto (en tanto por 1).

Su implementación fue la siguiente:

En primer lugar, se dividieron los espectros contenidos en la base de datos refinada en los sets de entrenamiento (80 % de los espectros) y test (20 % de los espectros) de forma aleatoria. En segundo lugar, se utilizó el set de entrenamiento para generar un clasificador basado en un algoritmo de Machine Learning con la combinación óptima de hiperparámetros. En tercer lugar, el clasificador fue utilizado para generar predicciones del set de entrenamiento y test. En cuarto lugar, se repitieron los pasos anteriores 20 veces (i.e. se repitió 20 veces la obtención de predicciones para diferentes sets de entrenamiento y test). En quinto lugar, se utilizaron los datos obtenidos por la repetición del tercer paso para obtener el número de veces que un espectro ha sido clasificado en cada clase, tratando de forma independiente las predicciones del set entrenamiento y test (i.e. se generaron 2 matrices de confusión desagregadas independientes, una para el set de entrenamiento y, otra para el set de test). En sexto lugar, se normalizaron los resultados del quinto paso sobre 1, obteniendo las matrices de confusión desagregadas para los sets de entrenamiento y test. En séptimo lugar, se utilizaron los datos obtenidos por la repetición del tercer paso para obtener el número de veces que los espectros de una clase han sido clasificados en cada clase, tratando de forma independiente las predicciones de los set entrenamiento y test (i.e. se generaron 2 matrices de confusión agregadas independientes, una para el set de entrenamiento y, otra para el set de test). En octavo lugar, se normalizaron los datos del séptimo paso sobre 1, obteniendo las matrices de confusión agregadas para los sets de entrenamiento y test.

5.2.4 Tiempos de clasificación

Los tiempos de clasificación para los diferentes algoritmos de Machine Learning y Deep Learning son medidos con la función *tic/toc* de Matlab®, que únicamente mide tiempos superiores a 100 ms con precisión. Sin embargo, los tiempos de clasificación están en el orden de 10 ms, lo que obliga a agrupar varios tiempos de clasificación en una única medida de la función *tic/toc* y, después realizar la media para obtener el tiempo de clasificación individual (como se sugiere en la ayuda de Matlab®).

La implementación de la medida de tiempos de clasificación fue la siguiente:

En primer lugar, se dividieron los espectros contenidos en la base de datos refinada en los sets de entrenamiento (80 % de los espectros) y test (20 % de los espectros) de forma aleatoria. En segundo lugar, se entrenó un clasificador en base a un algoritmo de Machine Learning con una combinación específica de hiperparámetros. En tercer lugar, el clasificador obtenido en el segundo paso se utilizó para clasificar n espectros EIS seleccionados de forma aleatoria. En cuarto lugar, se midió el tiempo que tarda en ejecutarse el tercer paso con la función *tic/toc*. Este debe ser superior a 10 ms, de lo contrario se aumenta n. En quinto lugar, se repitieron los pasos tercero y cuarto 10 000 veces (i.e. se midió 10 000 veces el tiempo que demora clasificar n espectros). En sexto lugar, se dividieron los tiempos obtenidos por n, obteniendo el tiempo medio en clasificar un espectro EIS.

Capítulo 6

Resultados

6.1 Machine Learning

6.1.1 Support Vector Machine

Optimización de hiperparámetros

La figura 6.1 muestra para cada tipo de Kernel (i.e. lineal, cuadrático, cúbico y gausiano) la exactitud media obtenida en el entrenamiento y test para los diferentes valores del Box Constraint y Kernel Scale.

Por un lado, respecto al Box Constraint: En los entrenamientos (i.e. las subfiguras 1), un mayor valor de Box Constraint aumenta la exactitud, ya que, aumenta la penalización de los errores cometidos. Sin embargo, para los test (i.e. las subfiguras 2) un mayor valor de Box Constrain no siempre aumenta la exactitud, ya que, valores elevados de Box Constrain propician el sobreajuste y, por lo tanto, la exactitud en el test disminuye a causa de esto.

Por otro lado, respecto al Kernel Scale: El set de entrenamiento y test presentan los valores más elevados de exactitud para valores de Kernel Scale cercanos a 1, lo que puede deberse a la normalización previa que realiza Matlab® de los datos de entrada.

En cuanto a la optimización de hiperparámetros: Por un lado, las subfiguras concernientes al set de entrenamiento muestran una diferencia significativa de colores (i.e. colores desde el azul blanquecino hasta azul propiamente dicho) y, por lo tanto, existen combinaciones de Kernel Scale y Box Constraint con resultados significativamente más favorables que otros, de un punto de vista de la exactitud.

Por otro lado, las subfiguras correspondientes al set de test muestran un estrecho margen de colores (i.e. un rojo blanquecino con ligeras variaciones), lo que indica una variación mínima de la exactitud. Sin embargo, dentro de la estrecha variabilidad es posible hallar el mayor valor de

exactitud media (39.41 %) para el Kernel gaussiano con un Box Constraint y Kernel Scale de 18.10 y 1.38, respectivamente.

Matriz de confusión

Las matrices de confusión agregadas para el algoritmo Support Vector Machine optimizado (i.e. Kernel Gaussiano, Box Constrain de 18.1 y Kernel Scale de 1.38) se pueden ver en la figura 6.2.

Para el set de entrenamiento, la figura 6.2a muestra la existencia de una fuerte confusión de clases, ya que, todas las clases son confundidas con la clase 1. Esta confusión de clases se acentúa para el set de test, como se muestra en la figura 6.2b, siendo las clases 3, 4 y 5 las de mayor ratio de confusión de clases.

Por lo tanto, el algoritmo de SVM optimizado utiliza como estrategia clasificar correctamente las clases 1 y 2 mayoritariamente, lo que permite explicar la exactitud media (39.41 %) del siguiente modo: Las clases 1 y 2 son las más abundantes en la base de datos con un 34.37 % y 22.24 % respectivamente y, por lo tanto, enfocarse en realizar una clasificación casi perfecta para la clase 1 y, complementarlo con algunas clasificaciones correctas del resto de clases (sobretodo de la clase 2) permite alcanzar unos porcentajes de exactitud ligeramente superiores al porcentaje de la clase más abundante.

Las matrices de confusión desagregadas para el algoritmo Support Vector Machine optimizado son mostradas en la figura 6.3. Para el set de entrenamiento (figura 6.3a) y test (figura 6.3b) se observa una confusión de todas las clase con la clase 1. Sin embargo, dicha confusión de clases es tan intensa que no permite relacionar su origen con la morfología de las imágenes.

Tiempo de clasificación

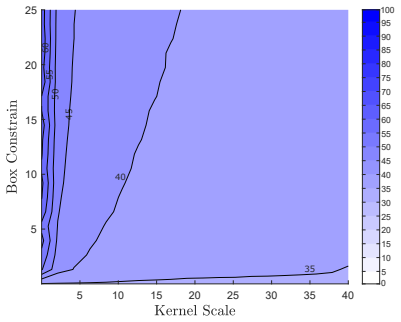
La distribución del tiempo de clasificación (i.e. el tiempo transcurrido en clasificar un ejemplo por el algoritmo ya entrenado) es mostrado en la figura 6.4. El tiempo medio es 7.79 y, los valores que engloban al 95 % de la población son 7.41 ms y 8.65 ms.

6.1.2 Neural Network

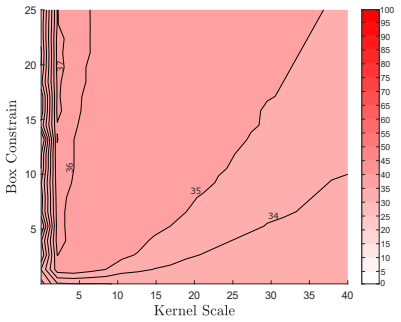
Optimización de hiperparámetros

Por un lado, en las subfiguras correspondientes al set de entrenamiento (i.e. las subfiguras 1 pertenecientes a la figura 6.5) se observa como un mayor número de neuronas produce una clasificación más acertada para el set de entrenamiento (pasando de una exactitud media de 45 % al 90 % aproximadamente). Puesto que, más neuronas implican más grados de libertad, lo que permite ajustar mejor los datos de entrenamiento.

Por otro lado, las subfiguras correspondientes al set de test (i.e. subfiguras 2 pertenecientes a la figura 6.5) presentan una exactitud media inferior al set de entrenamiento (alrededor del 40 %), lo que propicia que las barras engloben al porcentaje correspondiente de la clase más abundante (i.e. clase 1), lo que indica un posible sesgo de clasificación hacia la clase 1.

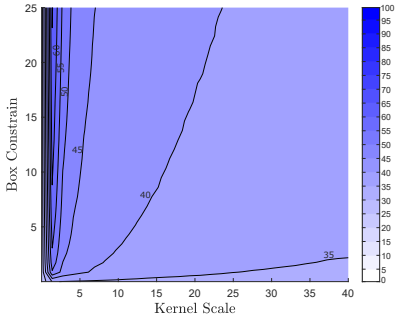


(a.1) Entrenamiento

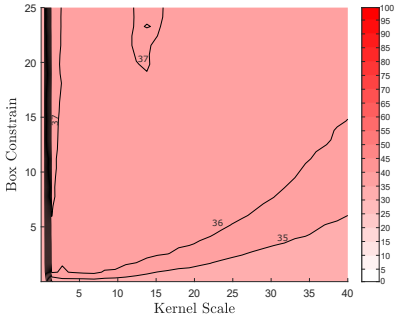


(a.2) Test

(a) Kernel lineal

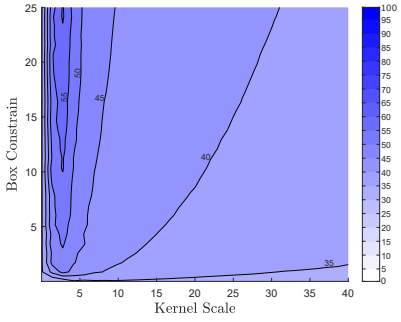


(b.1) Entrenamiento

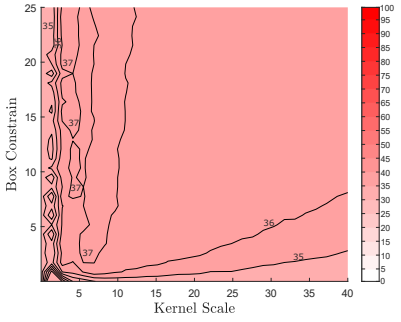


(b.2) Test

(b) Kernel cuadrático

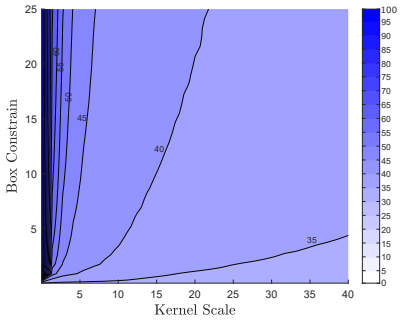


(c.1) Entrenamiento

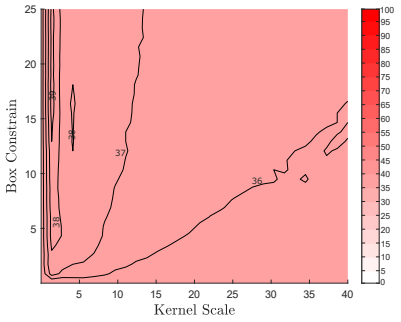


(c.2) Test

(c) Kernel cúbico



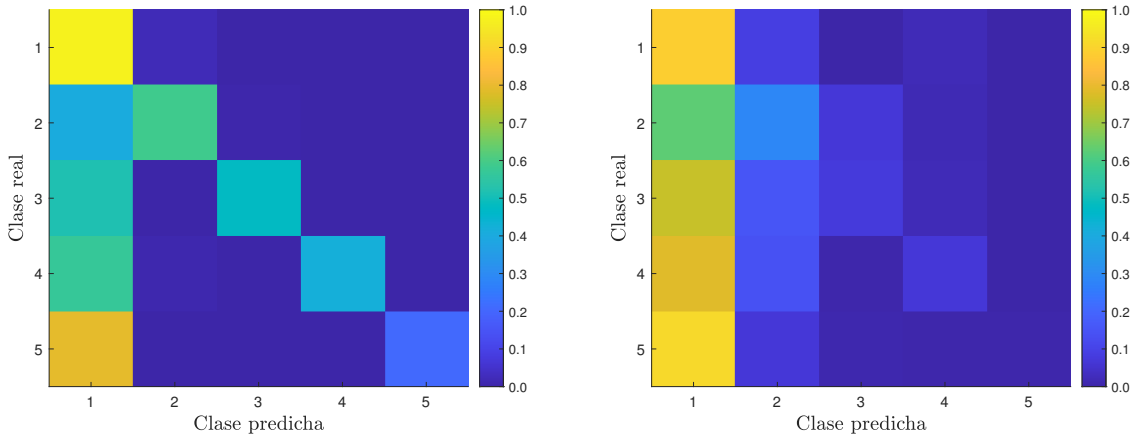
(d.1) Kernel gausiano



(d.2) Test

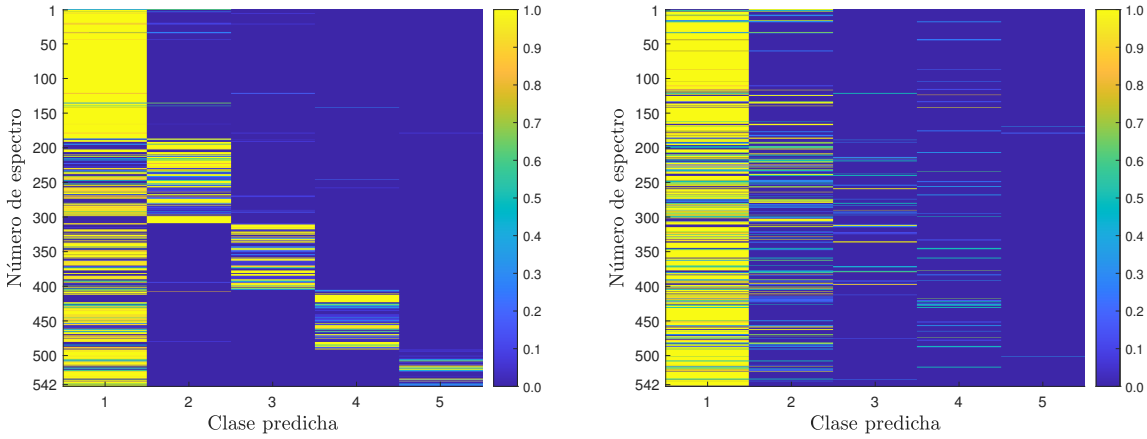
(d) Kernel Gaussiano

Figura 6.1: Exactitud media del algoritmo Support Vector Machine del set de entrenamiento (a.1, b.1, c.1 y d.1) y test (a.2, b.2, c.2 y d.2) para diferentes Kernels, Box Contrains y Kernel Scales. La exactitud media se ha calculado repitiendo 50 veces cada entrenamiento y test, excepto para el Kernel Lineal que se ha repetido 10 veces.



(a) Entrenamiento **(b)** Test

Figura 6.2: Matrices de confusión agregadas del algoritmo Support Vector Machine optimizado (i.e. Kernel Gaussian, Box Constrain de 18.1 y Kernel Scale de 1.38). Cada matriz de confusión se ha construido a partir del entrenamiento (a) y testeo (b) del algoritmo Support Vector Machine optimizado 20 veces. Por lo tanto, la magnitud representada en la escala de color es el porcentaje de veces en tanto por 1 que el algoritmo ha clasificado cierta clase como 1, 2, 3, 4 o 5.



(a) Entrenamiento **(b)** Test

Figura 6.3: Matrices de confusión desagregadas del algoritmo machine learning optimizado (i.e. Kernel Gaussian, Box Constrain de 18.1 y Kernel Scale de 1.38). Cada matriz de confusión se ha construido a partir del entrenamiento (a) y testeo (b) del algoritmo Support Vector Machine optimizado 20 veces. Por lo tanto, la magnitud representada en la escala de color es el porcentaje de veces en tanto por 1 que el algoritmo ha clasificado cierto espectro como clase 1, 2, 3, 4 o 5.

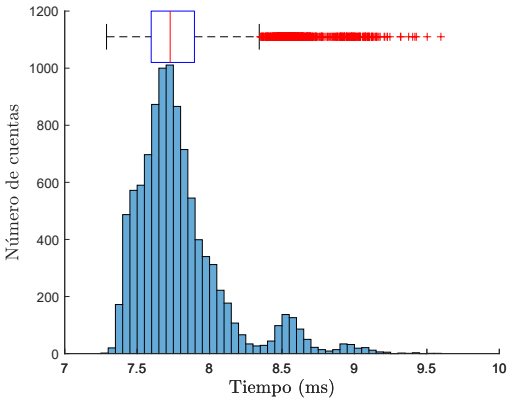


Figura 6.4: Distribución de tiempos de clasificación del algoritmo Support Vector Machine optimizado (i.e. Kernel Gaussian, Box Constrain de 18.1 y Kernel Scale de 1.38), para 10 000 repeticiones. La medida de tiempos de clasificación se ha realizado con un ordenador HP Pavilion Gaming Laptop 15-ec2xxx, con un procesador AMD Ryzen 5 con 12 CPUs y 3.3 GHz, una memoria RAM de 16 384 MB y una gráfica dedicada NVIDIA GeForce GTX 1650.

Además, la exactitud no mejora sustancialmente al aumentar el número de neuronas en la capa oculta, ya que, las barras de error están solapadas casi por completo (i.e. no hay una diferencia estadísticamente significativa).

Por lo tanto, no es posible la optimización del número de neuronas para cada función de activación en base a los resultados del test. Por ello, se optimiza el número de neuronas en función del set de entrenamiento (i.e. se elige el número de neuronas por encima del cual no se produce una mejora significativa en la exactitud de clasificación del set de entrenamiento), lo que resulta en 18 neuronas para las 3 funciones de activación.

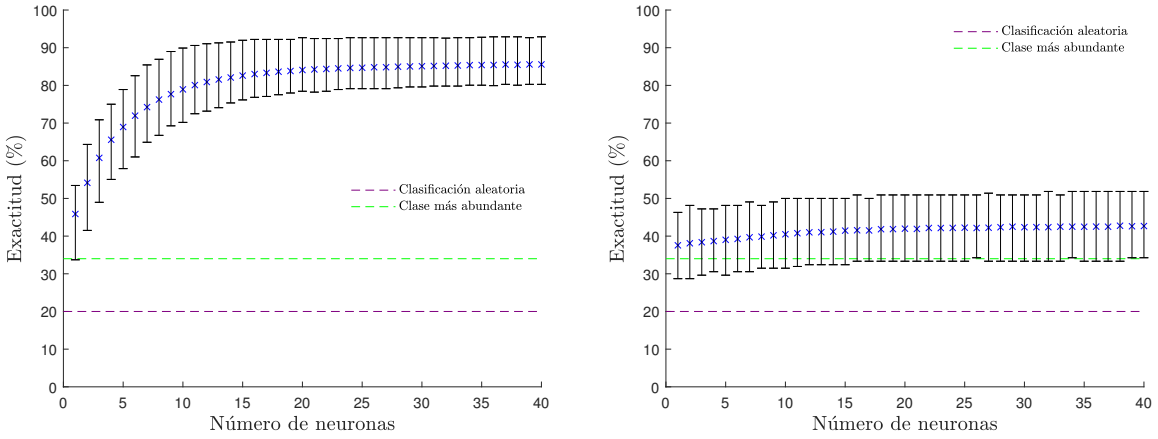
Los 3 modelos que se derivan de la optimización del número de neuronas (uno para cada función de activación) son comparados en la figura 6.6 donde, se observa que no existe una diferencia estadísticamente significativa entre ellos. Sin embargo, para evitar realizar cálculos redundantes se elige uno, el correspondiente con la función de activación ReLU y, se analizan las matrices de confusión y distribución de tiempos de clasificación con él.

Matrices de confusión

Las matrices de confusión agregadas para el algoritmo neural netowork optimizado (i.e. con 18 neuronas y la función de activación ReLU en la capa oculta) son mostradas en la figura 6.7 para el set de entrenamiento y test, respectivamente.

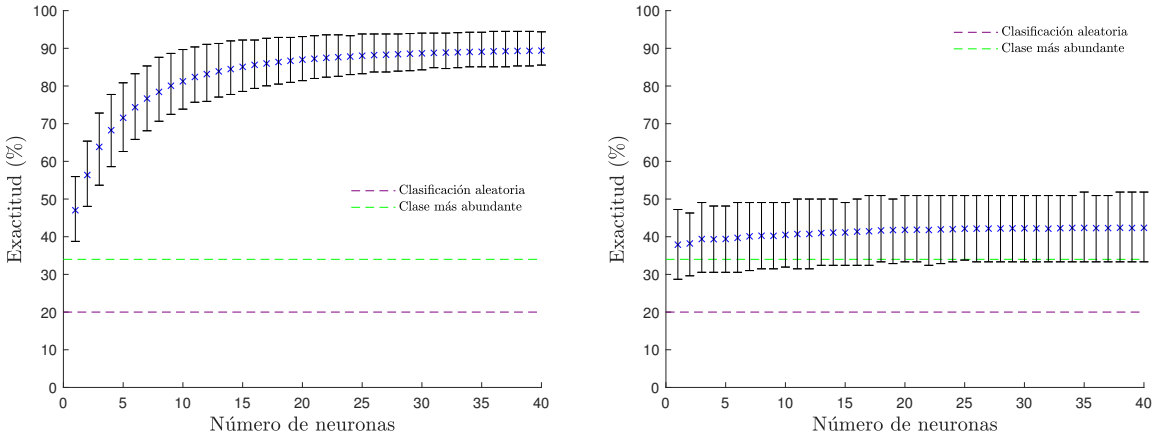
Por un lado, la figura 6.7a contiene valores cercanos a 1 en la diagonal y, por lo tanto, indica un clasificador capaz de clasificar un espectro satisfactoriamente en la mayoría de los casos, lo que concuerda con la elevada exactitud (i.e. aproximadamente del 80 %) mostrada en la figura 6.5a.1 para el set de entrenamiento. Sin embargo, muestra una ligera tendencia a clasificar espectros que no pertenecen a la clase 1 como clase 1, estando dicha tendencia acentuada para la clase 5.

Por otro lado, la figura 6.7b muestra menos clasificaciones correctas, ya que, los valores de la diagonal son inferiores a los mostrados en la figura 6.7a, lo que concuerda con los resultados mostrados en la figura 6.5a.2. Además, permite observar que el mayor ratio de acierto se da



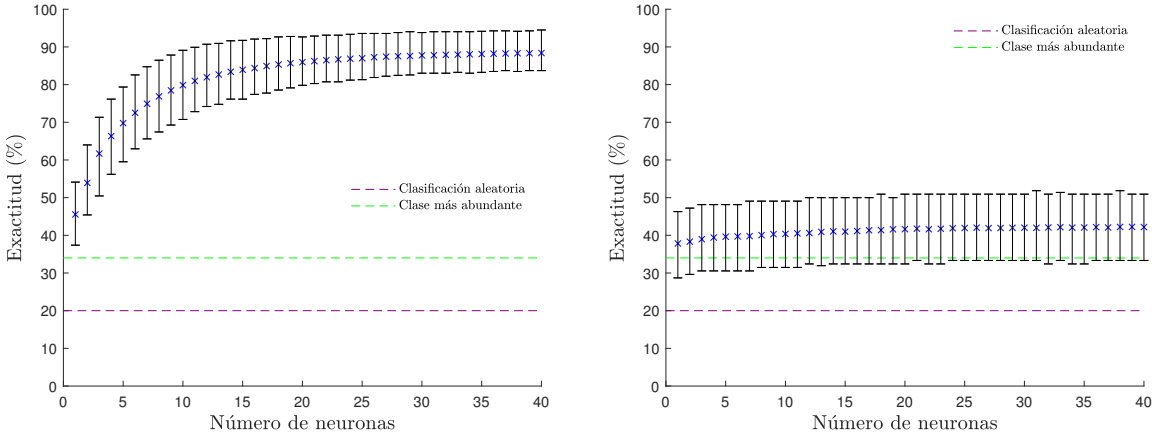
(a.1) Entrenamiento **(a.2)** Test

(a) Activación ReLU



(b.1) Entrenamiento **(b.2)** Test

(b) Activación tangente hiperbólica.



(c.1) Entrenamiento **(c.2)** Test

(c) Activación sigmoide

Figura 6.5: Exactitud media obtenida en el algoritmo Neural Network para los set de entrenamiento (a.1, b.1, c.1) y test (a.2, b.2, c.2) para diferentes funciones de activación y número de neuronas. Las barras de error para cada número de neuronas han sido calculadas repitiendo cada entrenamiento 2000 veces y, eligiendo los valores que engloban al 95 % de la población obtenida. Además, se muestra el porcentaje de acierto esperado para un clasificador aleatorio (morado), y un clasificador cuya predicción sea siempre la clase más abundante (verde).

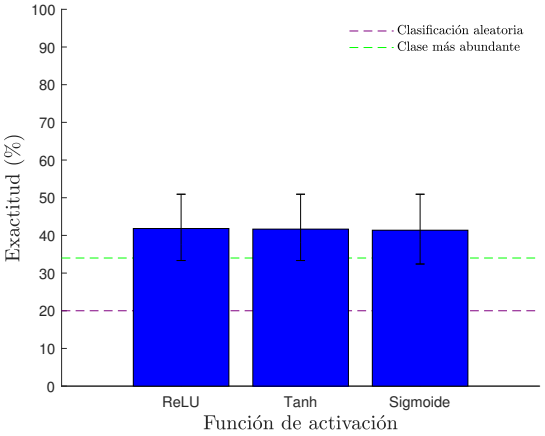


Figura 6.6: Exactitud del algoritmo neural con el número de neuronas optimizado (i.e. 18 neuronas) para diferentes funciones de activación. Las barras de error para cada función de activación han sido calculadas repitiendo cada entrenamiento 2000 veces y, eligiendo los valores que engloban al 95 % de la población obtenida. Además, se muestra la exactitud esperada para un clasificador aleatorio (morado) y, un clasificador cuya predicción sea siempre la clase más abundante (verde).

para la clase 1 y, una tendencia a confundir las clases 2, 3, 4 y 5 con la clase 1. Por lo tanto, la proximidad de las barras de error de la figura 6.5a.2 con el porcentaje de la clase más abundante es consecuencia de la clasificación de gran parte de los espectros como clase 1.

Las matrices de confusión desagregadas para el algoritmo Neural Network optimizado (i.e con 18 neuronas y la función de activación ReLU para la capa de activación) para los sets de entrenamiento y test son mostradas en las figuras 6.8a y 6.8b, respectivamente.

En las figuras 6.8a y 6.8b se puede notar como los espectros 523-529, mostrados en la figura 6.9, pertenecientes a la clase 5 son clasificados incorrectamente como clase 1, cuyos espectros representativos son mostrados en la figura 6.10.

Los espectros mostrados en la figura 6.9 y 6.10 comparten como característica común que su forma está compuesta por una semicircunferencia y/o una línea recta, lo que explica los elevados ratios de confusión entre las clases 1 y 5 para estos espectros.

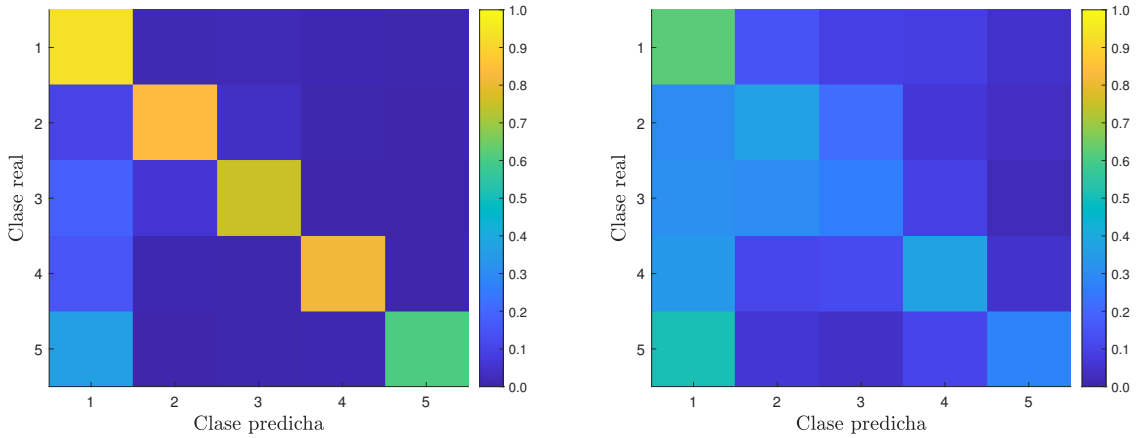
Tiempo de clasificación

La distribución del tiempo de clasificación (i.e. el tiempo transcurrido en clasificar un ejemplo por el algoritmo ya entrenado) es mostrado en la figura 6.11. El tiempo medio es 4.16 ms y, los valores que engloban al 95 % de la población son 4.07 ms y 4.25 ms.

6.1.3 Naive Bayes

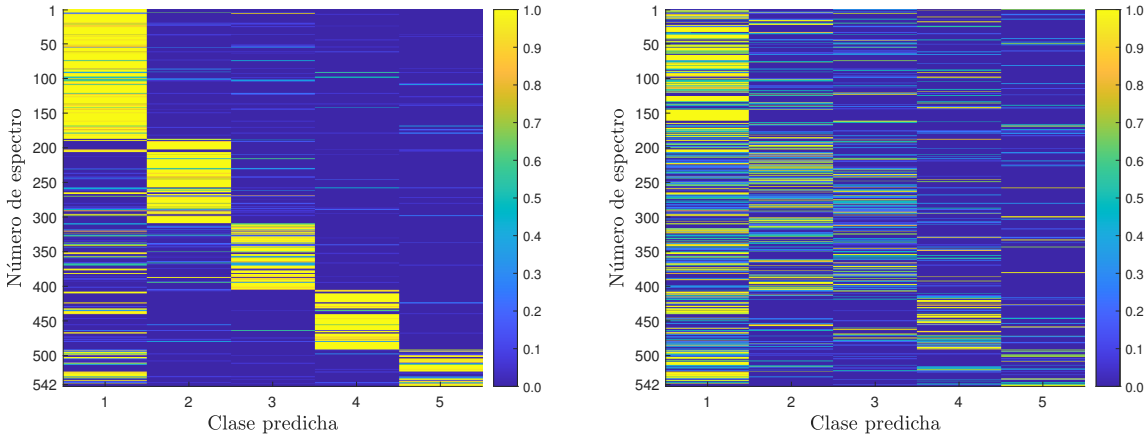
Optimización del hiperparámetro

En la figura 6.12a y 6.12b se observa que la exactitud media para el set de datos de entrenamiento y test está por debajo de la clasificación aleatoria (i.e. inferior al 20 %). Por lo tanto, una clasificación completamente aleatoria tendría mejores resultados que el algoritmo NB con cualquiera de las distribuciones estudiadas.



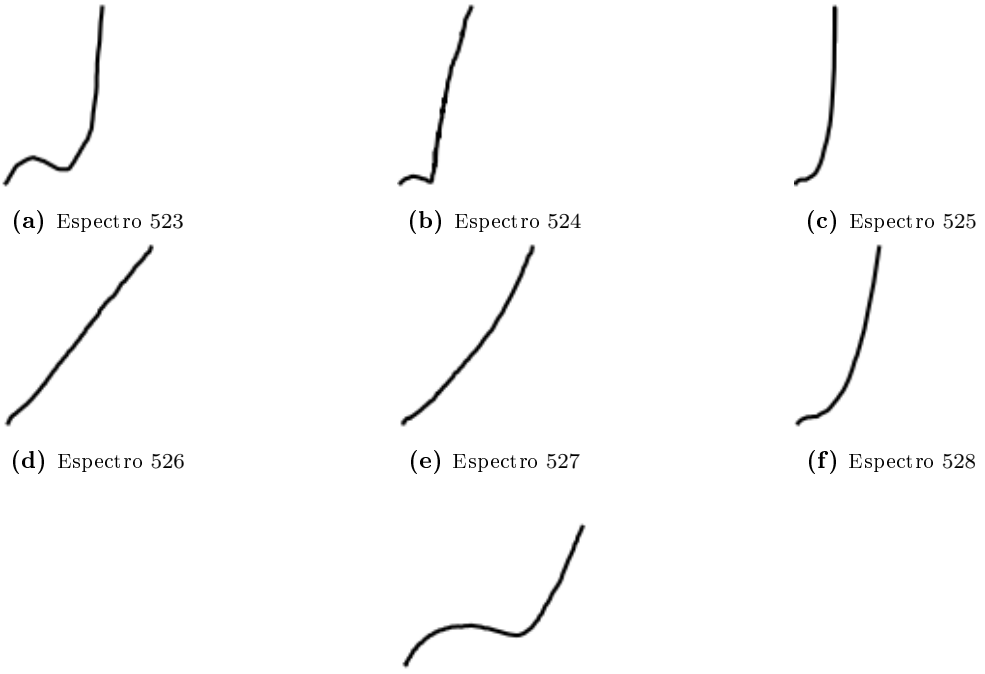
(a) Datos entrenamiento **(b)** Datos test

Figura 6.7: Matrices de confusión agregadas del algoritmo Neural Network optimizado (i.e. 18 neuronas y función de activación ReLU en la capa oculta). Cada matriz de confusión se ha construido a partir del entrenamiento (a) y testeo (b) del algoritmo Neural Network optimizado 20 veces. Por lo tanto, la magnitud representada en la escala de color es el porcentaje de veces en tanto por 1 que el algoritmo ha clasificado cierta clase como 1, 2, 3, 4 o 5.



(a) Datos entrenamiento **(b)** Datos test

Figura 6.8: Matrices de confusión desagregadas del algoritmo Neural Network optimizado (i.e. 18 neuronas y función de activación ReLU en la capa oculta). Cada matriz de confusión se ha construido a partir del entrenamiento (a) y testeo (b) del algoritmo Neural Network optimizado 20 veces. Por lo tanto, la magnitud representada en la escala de color es el porcentaje de veces en tanto por 1 que el algoritmo ha clasificado cierto espectro como clase 1, 2, 3, 4 o 5.



(a) Espectro 523 **(b)** Espectro 524 **(c)** Espectro 525

(d) Espectro 526 **(e)** Espectro 527 **(f)** Espectro 528

(g) Espectro 529

Figura 6.9: Espectros pertenecientes a la clase 5 clasificados sistemáticamente como clase 1 por el algoritmo Neural Network con 18 neuronas en la capa oculta y la función de activación ReLU.

(a) Espectro 20 **(b)** Espectro 171 **(c)** Espectro 178

Figura 6.10: Espectros representativos de la clase 1

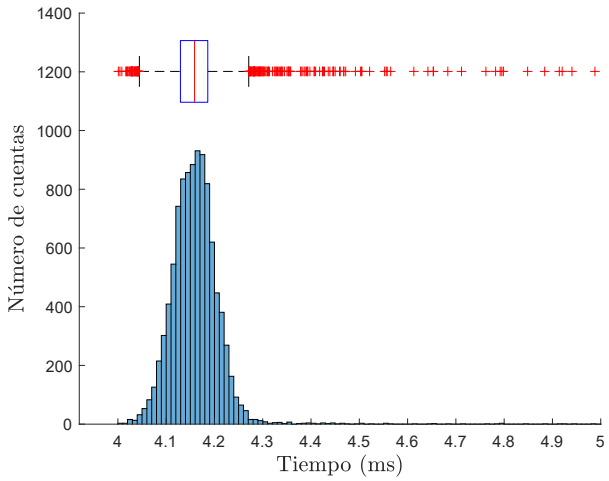
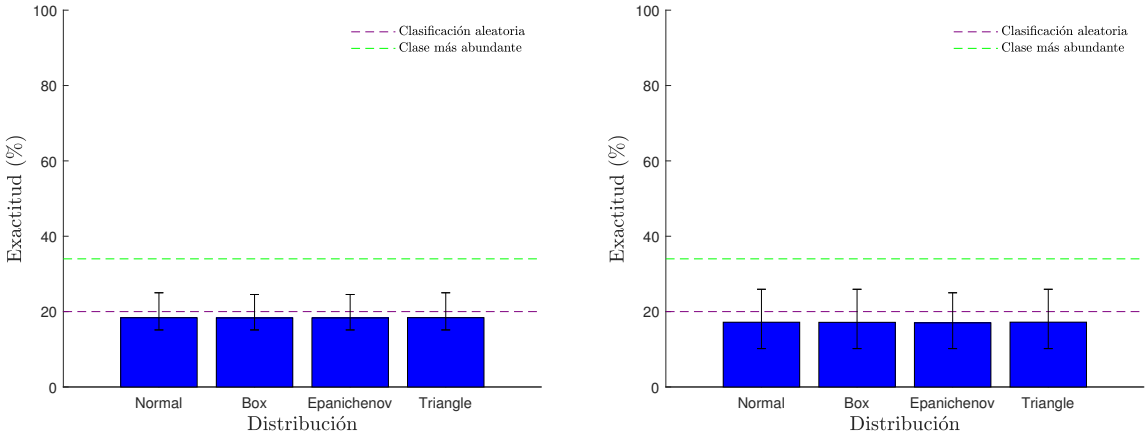
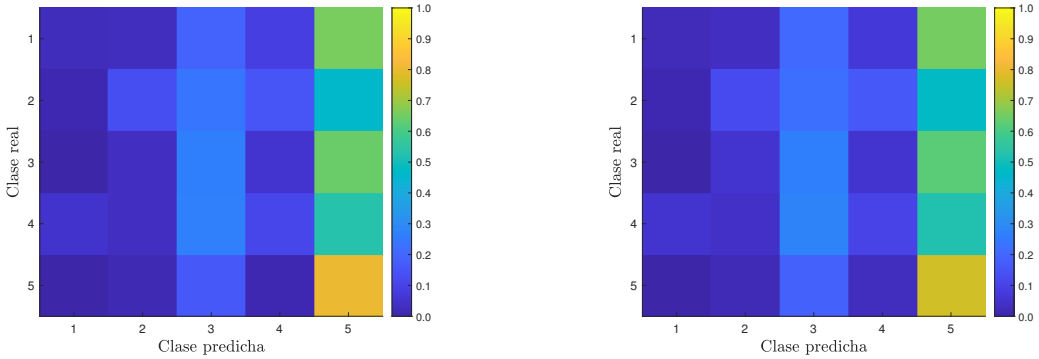


Figura 6.11: Distribución de tiempos de clasificación del algoritmo Neural Network optimizado (i.e. 18 neuronas y función de activación ReLU en la capa oculta), para 10 000 repeticiones. La medida de tiempos de clasificación se ha realizado con un ordenador HP Pavilion Gaming Laptop 15-ec2xxx, con un procesador AMD Ryzen 5 con 12 CPUs y 3.3 GHz, una memoria RAM de 16 384 MB y una gráfica dedicada NVIDIA GeForce GTX 1650.



(a) Entrenamiento **(b)** Test

Figura 6.12: Exactitud para el algoritmo Naive Bayes entre los set de entrenamiento y test para diferentes distribuciones. Las barras de error para cada distribucón han sido calculadas repitiendo cada entrenamiento 2000 veces y, eligiendo los valores que engloban al 95 % de la población obtenida. Además, se muestra la exactitud esperada para un clasificador aleatorio (morado) y, un clasificador cuya predicción sea siempre la clase más abundante (verde).



(a) Entrenamiento **(b)** Test

Figura 6.13: Matrices de confusión agregadas del algoritmo Naive Bayes optimizado (i.e. con la distribución normal). Cada matriz de confusión se ha construido a partir del entrenamiento y testeo del algoritmo Naive Bayes optimizado 20 veces. Por lo tanto, la magnitud representada en la escala de color es el porcentaje de veces en tanto por 1 que el algoritmo ha clasificado cierta clase como 1, 2, 3, 4 o 5.

En cuanto a las distribuciones estudiadas no se aprecia una diferencia estadísticamente significativa entre ellas, ya que, sus barras de error se solapan prácticamente por completo. Por lo tanto, no es posible la elección de una distribución que maximice la exactitud. Sin embargo, se escoge un distribución entre las 4 estudiadas, en este caso la distribución normal, y se obtienen las matrices de confusión y distribución de tiempos de clasificación asociados.

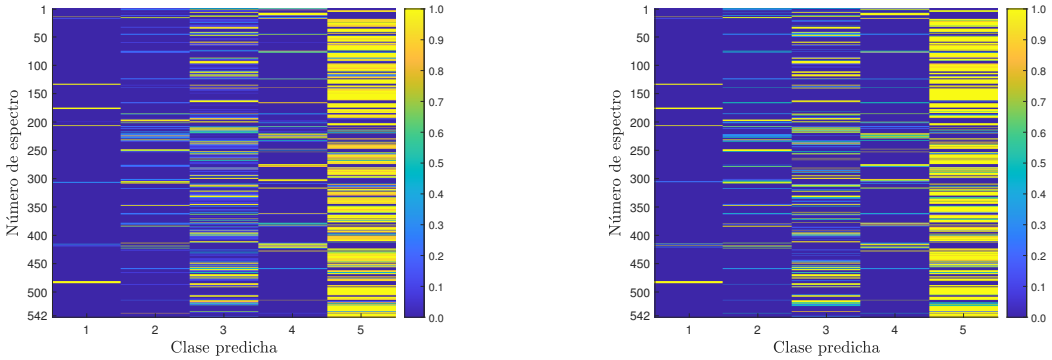
Matrices de confusión

Por un lado, las matrices de confusión agregadas mostradas en la figura 6.13 para los datos de entrenamiento y test permiten observar una tendencia a clasificar cualquier espectro como clase 5 indiscriminadamente, lo que explica los bajos porcentajes de exactitud mostrados en las figuras 6.12a y 6.12b, ya que, la clase 5 es la menos abundante.

Por otro lado, las matrices de confusión desagregadas mostradas en la figura 6.14 para los set de entrenamiento y test, permiten observar espectro por espectro la abundancia de clasificaciones erróneas de espectros pertenecientes a la clase 1, 2, 3 y 4 como clase 5. Sin embargo, mirar la confusión de clases espectro por espectro no permite identificar un patrón visual que explique esta confusión tan marcada de todas las clases con la clase 5.

Tiempos de clasificación

La distribución del tiempo de clasificación (i.e. el tiempo transcurrido en clasificar un ejemplo por el algoritmo ya entrenado) es mostrado en la figura 6.15. El tiempo de medio es 5.65 ms y, los valores que engloban al 95 % de la población son 5.50 ms y 5.86 ms.



(a) Entrenamiento **(b)** Test

Figura 6.14: Matrices de confusión desagregadas para cada imagen del algoritmo Naive Bayes optimizado (i.e. con la distribución normal). Cada matriz de confusión se ha construido a partir del entrenamiento y testeo del algoritmo Naive Bayes optimizado 20 veces. Por lo tanto, la magnitud representada en la escala de color es el porcentaje de veces en tanto por 1 que el algoritmo ha clasificado cierto espectro como clase 1, 2, 3, 4 o 5.

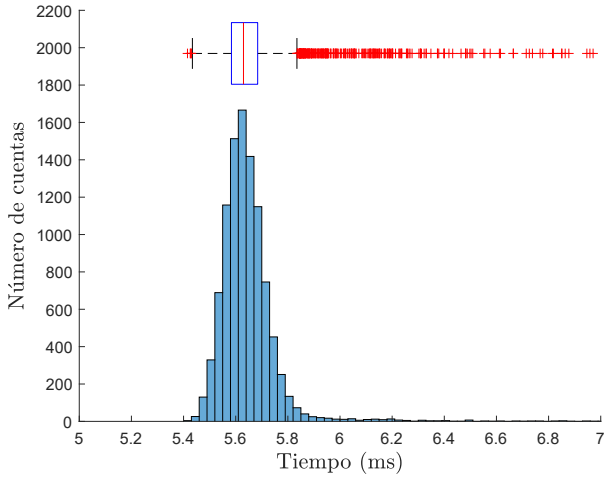
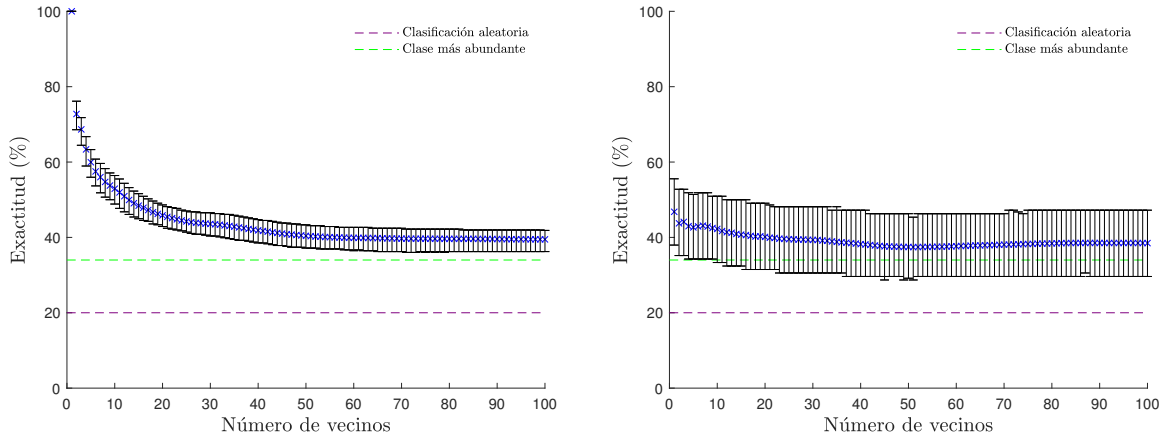
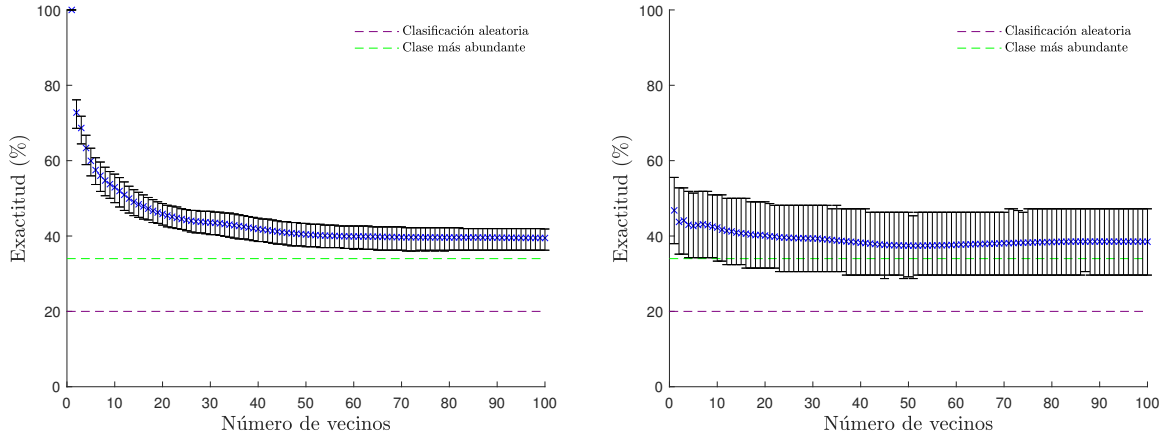


Figura 6.15: Distribución de tiempos de clasificación del algoritmo Naive Bayes optimizado (i.e. con la distribución normal), para 10 000 repeticiones. La medida de tiempos de clasificación se ha realizado con un ordenador HP Pavilion Gaming Laptop 15-ec2xxx, con un procesador AMD Ryzen 5 con 12 CPUs y 3.3 GHz, una memoria RAM de 16 384 MB y una gráfica dedicada NVIDIA GeForce GTX 1650.



(a.1) Entrenamiento **(a.2)** Test

(a) Distancia Euclídea



(b.1) Entrenamiento **(b.2)** Test

(b) Distancia de Minkowski

Figura 6.16: Exactitud para el algoritmo K Nearest Neighbor entre los sets de entrenamiento y test para la distancia Euclídea y de Minkowski y diferente número de vecinos. Las barras de error para cada número de divisiones han sido calculadas repitiendo cada entrenamiento 2000 veces y, eligiendo los valores que engloban al 95 % de la población obtenida. Además, se muestra la exactitud esperada para un clasificador aleatorio (morado) y, un clasificador cuya predicción sea siempre la clase más abundante (verde).

6.1.4 K Nearest Neighbor

Optimización de hiperparámetros

La distancia Euclidiana y de Minkowski, obtienen el mismo resultado de exactitud, como se muestra en la figura 6.16. Por lo tanto, las formas de medir la distancia seleccionadas no influyen en la exactitud del clasificador. Por ello, se selecciona una distancia, la Euclidiana en este caso, y se optimiza el número de vecinos.

El número de vecinos utilizados sí que tiene un gran impacto en la exactitud del clasificador, como se muestra en las figura 6.16a. Por un lado, en la figura 6.16a.1, se observa una disminución en la exactitud del clasificador para el set de entrenamiento con el aumento del número de vecinos seleccionados para realizar la clasificación (pasando de un 70 % al 40 % aproximadamente). Esto es consecuencia de que un número demasiado elevado de vecinos desvirtúa la clasificación, puesto

que, se tiene una elevada posibilidad de tomar observaciones pertenecientes a otras clases en consideración.

En el caso límite (i.e. un número de vecinos suficientemente grande) la exactitud tiende al porcentaje de la clase más abundante para un número de vecinos alto (i.e. por encima de 70 vecinos), ya que, aumenta la posibilidad de seleccionar vecinos pertenecientes a otras clases, siendo la clase más abundante la más probable para ser seleccionada.

Otro fenómeno significativo es que la longitud de las barras de error de la exactitud de entrenamiento para 1 vecino es 0 y, por lo tanto, siempre se obtiene un 100 % de exactitud sin importar cuantas veces se repita el experimento. Esto es debido a que el algoritmo KNN guarda la posición y clase de las observaciones de entrenamiento. Por ello, al intentar clasificar una observación utilizada para el entrenamiento, la observación más cercana será siempre esa misma observación y, como solo se utiliza 1 vecino siempre se clasifica correctamente.

Por otro lado, la figura 6.16a.2 muestra la independencia del set de test con número de vecinos, ya que, la exactitud media se mantiene prácticamente constante al rededor del 40 % y, las barras de error están solapadas casi por completo. Sin embargo, para 1 vecino es posible apreciar una mejora estadísticamente significativa en la exactitud y, por lo tanto, se toma 1 vecino como el número de vecinos óptimo.

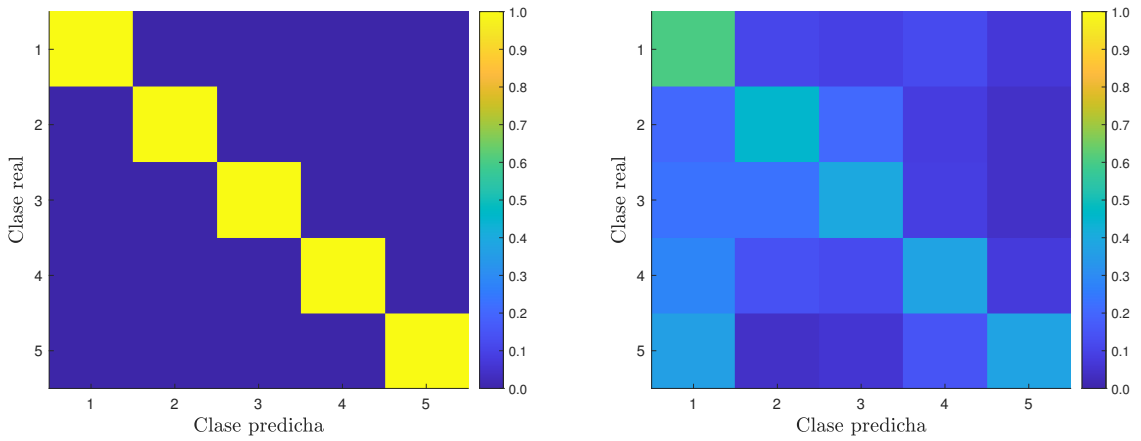
Matrices de confusión

Las matrices de confusión agregadas para el algoritmo K Nearest Neighbor optimizado (i.e. distancia Euclídea y 1 vecino) son mostradas en la figura 6.17 para entrenamiento y test. Por un lado, en la figura 6.17a se observa como el algoritmo KNN realiza una clasificación perfecta para el set de entrenamiento, ya que, se tienen valores de 1 en la diagonal y 0 para el resto de posiciones, lo que es coherente con la figura 6.16a.1.

Por otro lado, en la figura 6.17b se observa como el algoritmo KNN tiene tendencia a clasificar cualquier modelo como clase 1. Esto es coherente con que la clase 1 es la más abundante y, por lo tanto, lo más probable es que los vecinos más cercanos a cualquier observación pertenezcan a la clase 1.

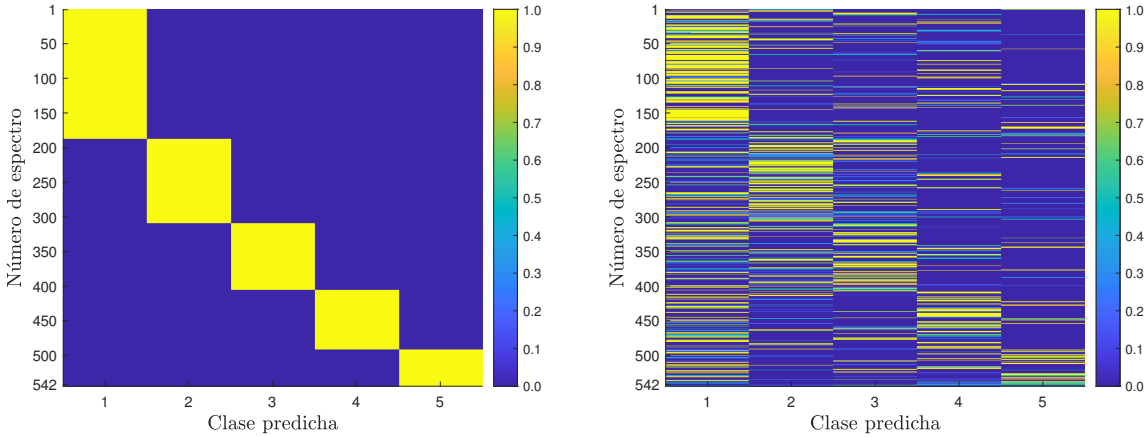
Las matrices de confusión desagregadas para el algoritmo K Nearest Neighbor optimizado (i.e. distancia Euclídea y 1 vecino) son mostradas en la figura 6.18 para los sets de entrenamiento y test. Por un lado, la figura 6.18a muestra una clasificación perfecta de cada espectro para el set de entrenamiento, lo que es coherente con el 100 % de exactitud mostrado en la figura 6.16a.1 y con la disposición diagonal de la matriz agregada mostrada en la figura 6.17a.

Por otro lado, la figura 6.18b muestra una tendencia a clasificar la mayoría de espectros como clase 1. Esta tendencia es tan pronunciada que es imposible detectar un patrón visual capaz de explicar el origen de la confusión de las clases 2, 3, 4 y 5 con la clase 1.



(a) Entrenamiento **(b)** Test

Figura 6.17: Matrices de confusión agregadas del algoritmo K Nearest Neighbor optimizado (i.e. con 1 vecino). Cada matriz de confusión se ha construido a partir del entrenamiento (a) y testeo (b) del algoritmo K Nearest Neighbor optimizado 20 veces. Por lo tanto, la magnitud representada en la escala de color es el porcentaje de veces en tanto por 1 que el algoritmo ha clasificado cierta clase como 1, 2, 3, 4 o 5.



(a) Entrenamiento **(b)** Test

Figura 6.18: Matrices de confusión desagregadas del algoritmo K Nearest Neighbor optimizado (i.e. 1 vecino). Cada matriz de confusión se ha construido a partir del entrenamiento (a) y testeo (b) del algoritmo K Nearest Neighbor optimizado 20 veces. Por lo tanto, la magnitud representada en la escala de color es el porcentaje de veces en tanto por 1 que el algoritmo ha clasificado cierto espectro como clase 1, 2, 3, 4 o 5.

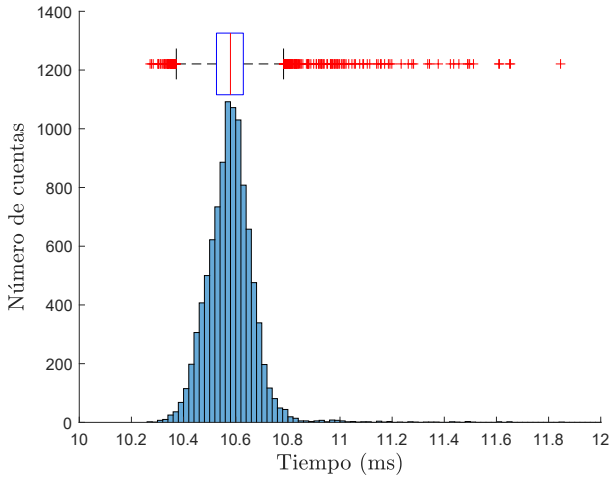


Figura 6.19: Distribución de tiempos de clasificación del algoritmo K Nearest Neighbor optimizado (i.e. 1 vecino), para 10 000 repeticiones. La medida de tiempos de clasificación se ha realizado con un ordenador HP Pavilion Gaming Laptop 15-ec2xxx, con un procesador AMD Ryzen 5 con 12 CPUs y 3.3 GHz, una memoria RAM de 16 384 MB y una gráfica dedicada NVIDIA GeForce GTX 1650.

Tiempos de clasificación

La distribución del tiempo de clasificación (i.e. el tiempo transcurrido en clasificar un ejemplo por el algoritmo ya entrenado) es mostrado en la figura 6.19. El tiempo medio es 10.56 ms y, los valores que engloban al 95 % de la población son 10.38 ms y 10.73 ms.

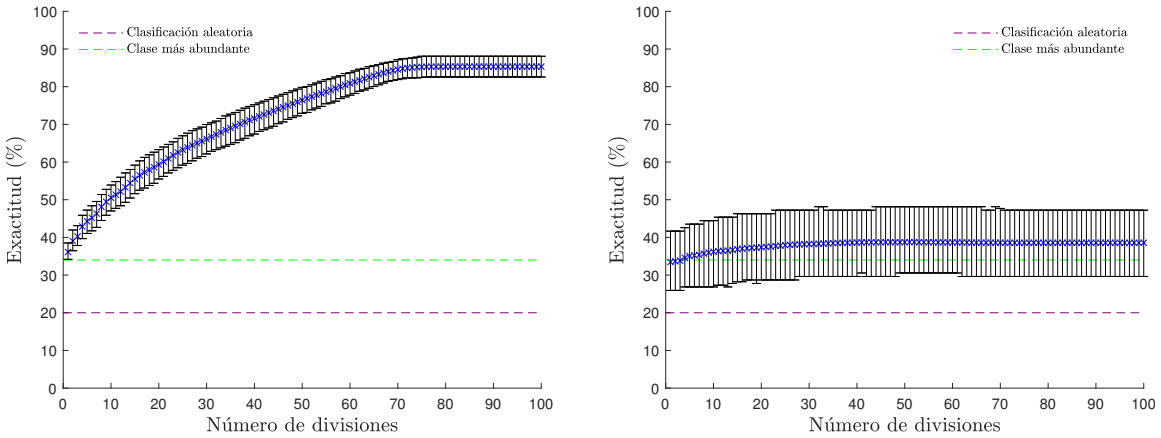
6.1.5 Decision Tree

Optimización de hiperparámetros

Por un lado, en la figura 6.20a se puede observar como un mayor número de divisiones propicia una mejor clasificación para el set de entrenamiento. Esto se debe al aumento de complejidad del árbol con el aumento del número de divisiones, lo que permite realizar clasificaciones más complejas. Sin embargo, por encima de 70 divisiones no se observa una mejora en la exactitud del set de entrenamiento y, por consiguiente, un aumento indefinido en la complejidad del árbol de decisión, no es suficiente para captar todos los matices del problema de clasificación planteado.

Por otro lado, en la figura 6.20b se puede observar como un aumento en el número de divisiones, apenas repercute de forma estadísticamente significativa en una mejor clasificación del set de test. Además, las barras de error están solapadas con el porcentaje de la clase más abundante y, por lo tanto, es posible que el algoritmo DT genere sistemáticamente modelos con tendencia a clasificar cualquier espectro como la clase más abundante (i.e. clase 1).

El solapamiento de las barras de error en la figura 6.20b, no permite hallar el número de divisiones que maximiza la exactitud en el set de test. Por ello, se elige el número de divisiones que maximiza la exactitud en el set de entrenamiento que se corresponde con 70 divisiones, puesto que, por encima de 70 divisiones no se observa una mejora en la exactitud.



(a) Entrenamiento **(b)** Test

Figura 6.20: Exactitud para el algoritmo Decision Tree entre el set de entrenamiento y test para diferentes números de divisiones. Las barras de error para cada número de divisiones han sido calculadas repitiendo cada entrenamiento 2000 veces y, eligiendo los valores que engloban al 95 % de la población obtenida. Además, se muestra la exactitud esperada para un clasificador aleatorio (morado) y, un clasificador cuya predicción sea siempre la clase más abundante (verde).

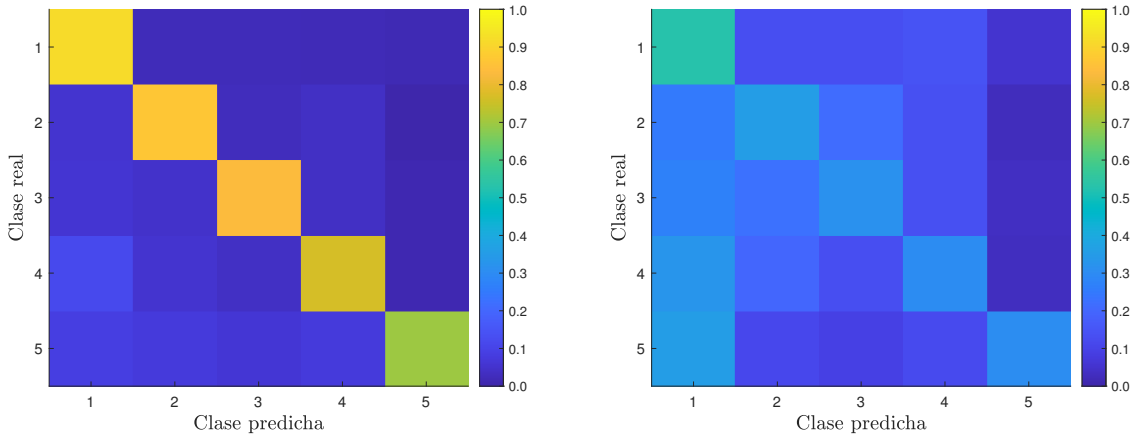
Matrices de confusión

Las matrices de confusión agregadas para el algoritmo Decision Tree optimizado (i.e. con 70 divisiones) son mostradas en la figura 6.21 para el set de entrenamiento y test.

Por un lado, la figura 6.21a muestra como el algoritmo Decision Tree con el número de divisiones óptimas (i.e. 70 divisiones) es capaz de predecir la clase de los espectros del set de entrenamiento con una elevada exactitud, ya que, los valores más altos se dan en la diagonal de la matriz. Aunque, el desempeño para la clasificación de la clase 5 es sensiblemente inferior.

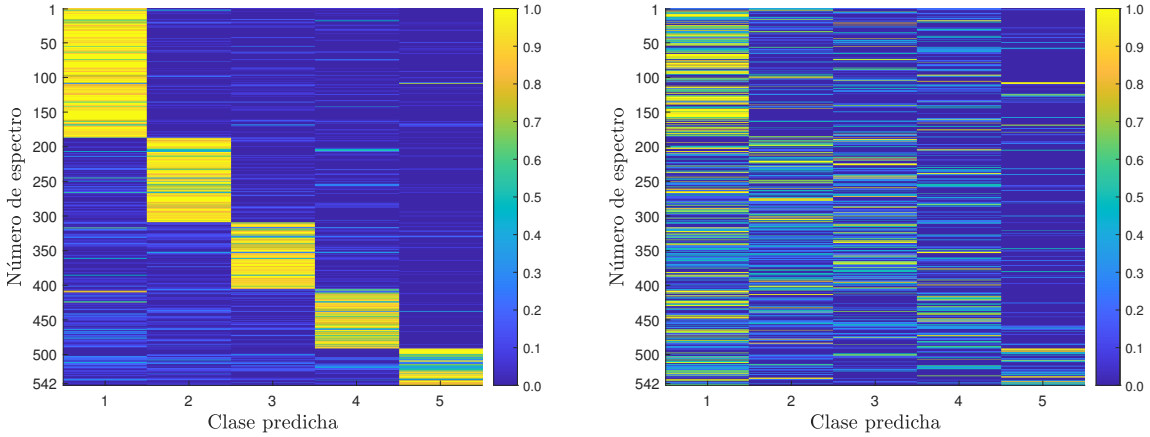
Por otro lado, la figura 6.21b muestra como el algoritmo Decision Tree con el número de divisiones óptimas (i.e 70 divisiones), no realiza una clasificación correcta para la mayoría de espectros, ya que la diagonal de la matriz presenta valores bajos comparados con los obtenidos en la figura 6.21a. Además, permite observar el sesgo de clasificación preferencial para la clase 1 dado que, la columna de la clase predicha 1 contiene valores relativamente altos sin importar la clase real.

Las matrices de confusión desagregadas para el algoritmo Decision Tree óptimo (i.e. con 70 divisiones) son mostradas en la figura 6.22 para los sets de entrenamiento y test, respectivamente. Por ejemplo, en ambas figuras el espectro 409, mostrado en la figura 6.23a, perteneciente a la clase 4 es clasificado incorrectamente como clase 1. En cambio, el espectro 464, mostrado en la figura 6.23b, que tiene una gran similitud visual con el espectro 409 es clasificado como clase 4 para el entrenamiento y como clase 2 para el test. Por lo tanto, el algoritmo decisión tree realiza la clasificación en base a criterios que no tienen una correspondencia visual clara.



(a) Entrenamiento **(b)** Test

Figura 6.21: Matrices de confusión agregadas del algoritmo Decision Tree optimizado (i.e. 70 divisiones). Cada matriz de confusión se ha construido a partir del entrenamiento (a) y testeo (b) del algoritmo Decision Tree optimizado 20 veces. Por lo tanto, la magnitud representada en la escala de color es el porcentaje de veces en tanto por 1 que el algoritmo ha clasificado cierta clase como 1, 2, 3, 4 o 5.



(a) Entrenamiento **(b)** Test

Figura 6.22: Matrices de confusión desagregadas del algoritmo Decision Tree optimizado (i.e. 70 divisiones). Cada matriz de confusión se ha construido a partir del entrenamiento (a) y testeo (b) del algoritmo Decision Tree optimizado 20 veces. Por lo tanto, la magnitud representada en la escala de color es el porcentaje de veces en tanto por 1 que el algoritmo ha clasificado cierto espectro como clase 1, 2, 3, 4 o 5.

(a) Espectro 409 **(b)** Espectro 464

Figura 6.23: Espectros pertenecientes a la clase 4 visualmente similares pero, clasificados de forma diferente por el algoritmo Decision Tree.

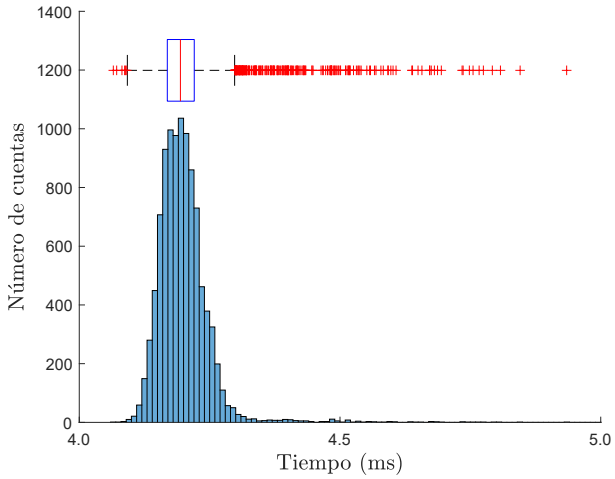


Figura 6.24: Distribución de tiempos de clasificación del algoritmo Decision Tree optimizado (i.e. 70 divisiones), para 10 000 repeticiones. La medida de tiempos de clasificación se ha realizado con un ordenador HP Pavilion Gaming Laptop 15-ec2xxx, con un procesador AMD Ryzen 5 con 12 CPUs y 3.3 GHz, una memoria RAM de 16 384 MB y una gráfica dedicada NVIDIA GeForce GTX 1650.

Tiempos de clasificación

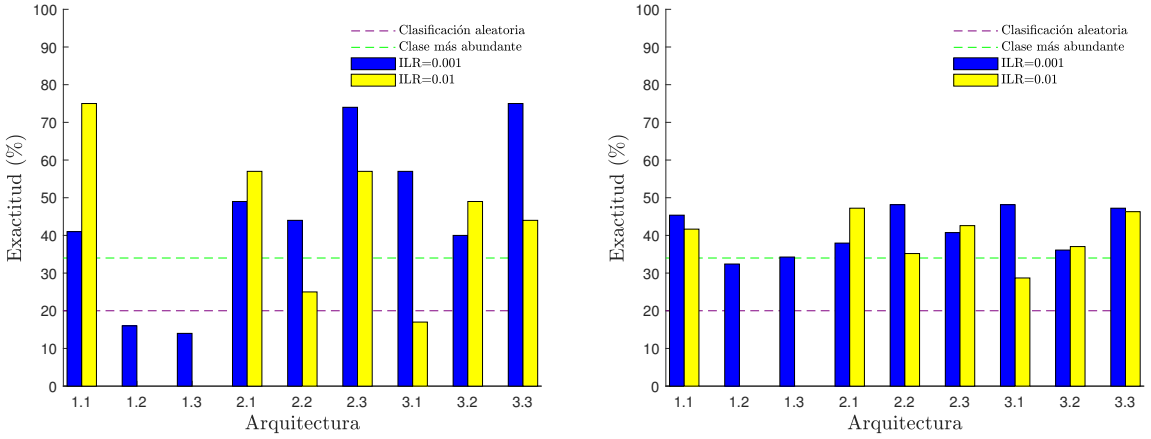
La distribución del tiempo de clasificación (i.e. el tiempo transcurrido en clasificar un ejemplo por el algoritmo ya entrenado) es mostrado en la figura 6.24. El tiempo medio es 4.20 ms y, los valores que engloban al 95 % de la población son 4.13 ms y 4.29 ms.

6.2 Deep Learning

6.2.1 Optimización de la parte convolucional

La figura 6.25 muestra los resultados del entrenamiento y test para diferentes combinaciones de arquitecturas e Initial Learn Rate (ILR). Sin embargo, no se muestran las 27 combinaciones posibles, ya que, algunas no convergen a una solución y el resultado final no es satisfactorio (i.e. un modelo en blanco que no puede predecir nada). Este fenómeno es propiciado por ILR demasiado altos, ya que, provocan cambios bruscos en la búsqueda de un óptimo (como se muestra en la figura 6.26b para los ILR=0.1 e ILR=0.01, donde a partir de un número de iteraciones la función no devuelve ningún valor para la pérdida), lo que explica que todas las combinaciones con ILR=0.1 sean descartadas.

También es posible observar una situación poco usual en la figura 6.25 para las arquitecturas 1.2 (con ILR=0.001), 1.3 (con ILR=0.001) y 3.1 (con ILR=0.01), una exactitud mayor en el test que en el entrenamiento. Esto es debido al comportamiento errático que presentan algunas arquitecturas, donde para una iteración la exactitud puede disminuir para el entrenamiento y aumentar para la validación y, en la siguiente iteración tener el comportamiento contrario (como se muestra en la figura 6.26), lo que posibilita que puntualmente la exactitud de la validación sea superior a la del test.



(a) Entrenamiento **(b)** Test

Figura 6.25: Exactitud de entrenamiento y test para diferentes combinaciones de arquitecturas de redes convolucionales e ILR (azul ILR=0.001, amarillo ILR=0.01). Además, se muestra la exactitud esperada para un clasificador aleatorio (morado) y, un clasificador cuya predicción sea siempre la clase más abundante (verde).

La figura 6.25a muestra como 2 de las 3 mejores arquitecturas (1.1, 2.3 y 3.3 con ILR de 0.01, 0.001 y 0.001, respectivamente) para el set de entrenamiento se corresponden con un ILR de 0.001. Además, la figura 6.25b confirma esta tendencia mostrando como las 3 mejores arquitecturas para el set de test (2.2, 3.1 y 3.3) tienen un ILR de 0.001, siendo estas elegidas como las mejores arquitecturas analizadas por ahora. Por lo tanto, se concluye que un ajuste más suave de los pesos en la red convolucional proporciona mejores resultados.

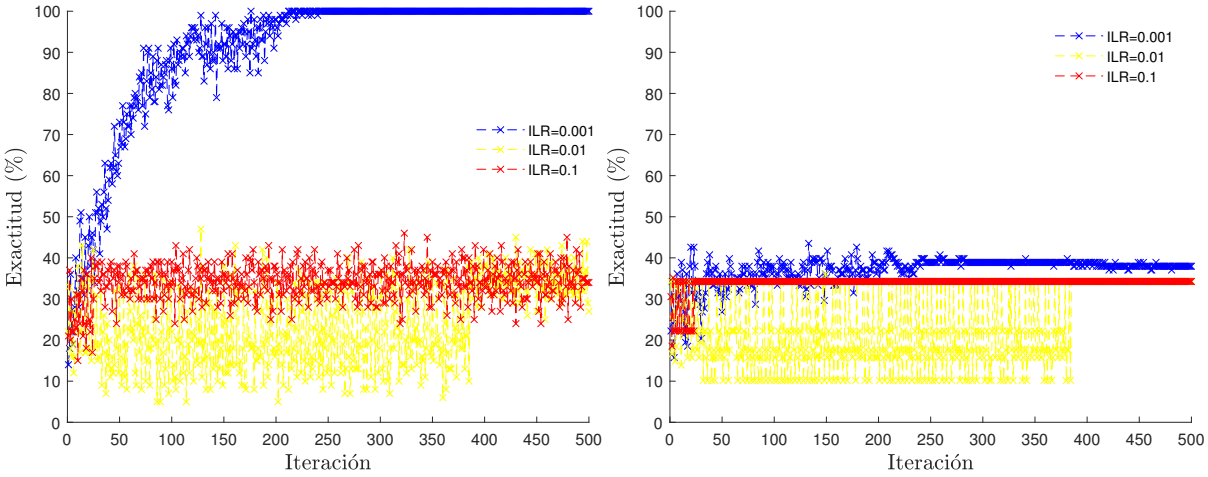
En cuanto a la bondad de la clasificación, la figura 6.25a muestra arquitecturas capaces de alcanzar en el entrenamiento exactitudes muy superiores a una clasificación exclusiva de la clase más abundante, llegando a sobrepasar el 70 %. Y la figura 6.25b muestra arquitecturas capaces de superar el 45 %. Aunque, estos resultados deben ser tomados con cautela, ya que, no se dispone de barras de error.

6.2.2 Optimización de la parte densa

La figura 6.27a muestra como la introducción de una capa neural extra disminuye de forma estadísticamente significativa la exactitud para el set de entrenamiento. Sin embargo, en la figura 6.27b apenas se aprecia dicha disminución. Esto indica que no existe más información por aprender de las imágenes utilizadas y, por ello, una capa extra de neuronas no ofrece mejores clasificaciones.

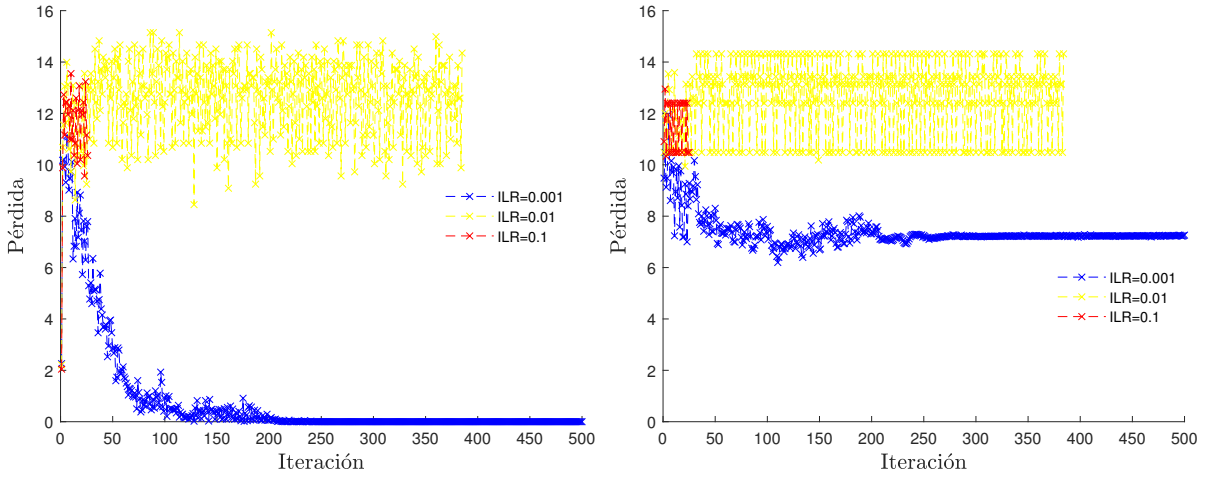
Las clasificaciones más pobres en el entrenamiento son debidas al número de iteraciones seleccionadas (i.e. el número de veces que se reajustarán los pesos), ya que, una capa extra exige más iteraciones para dar lugar a mejores clasificaciones en el entrenamiento. Sin embargo, se elige el número de iteraciones en base a minimizar el error de clasificación en el test y no en el entrenamiento para evitar el fenómeno del overfitting, lo que explica el comportamiento observado.

En cuanto a la elección de la mejor arquitectura, se realiza en base a los resultados del test mostrados en la figura 6.27b, donde se observa como la arquitectura 3.3.1 presenta la mejor exactitud (i.e una media de 42.27 %). Además, su límite inferior de error esta ligeramente por encima de la clase más abundante y, por lo tanto, la clasificación realizada corresponde a una



(a.1) Entrenamiento **(a.2)** Validación

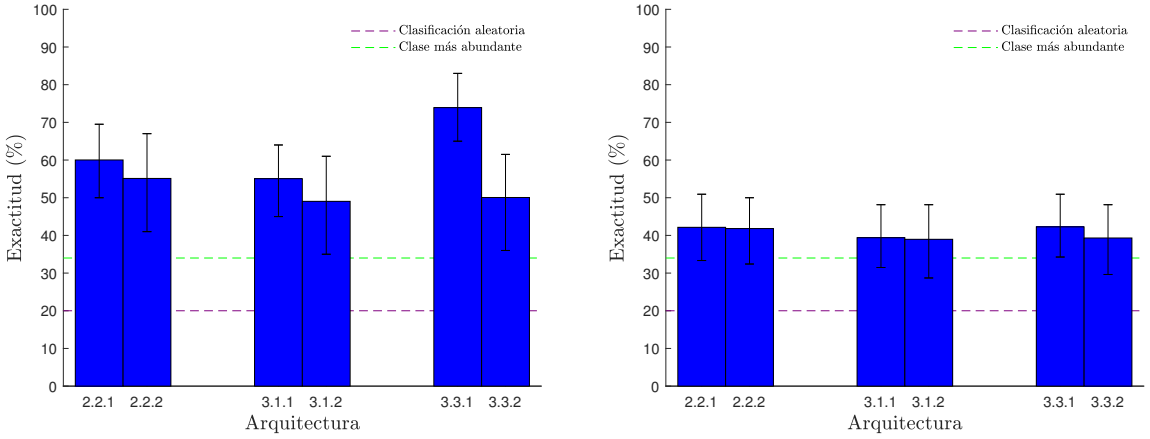
(a) Exactitud



(b.1) Entrenamiento **(b.2)** Validación

(b) Pérdida

Figura 6.26: Influencia de diferentes valores de Initial Learning Rate (ILR) (0.001, 0.01 y 0.1 en azul, amarillo y rojo, respectivamente) sobre la exactitud (a.1 y a.2) y pérdida (b.1 y b.2) de la arquitectura convolucional 1.2 durante el entrenamiento.



(a) Entrenamiento **(b)** Test

Figura 6.27: Exactitud para las mejores arquitecturas de redes convolucionales entre el set de entrenamiento y test. Las barras de error para cada arquitectura han sido calculadas repitiendo cada entrenamiento 2000 veces y, eligiendo los valores que engloban al 95 % de la población obtenida. Además, se muestra la exactitud esperada para un clasificador aleatorio (morado) y, un clasificador cuya predicción sea siempre la clase más abundante (verde).

estrategia más compleja que una simple clasificación de todos los espectros como la clase más abundante.

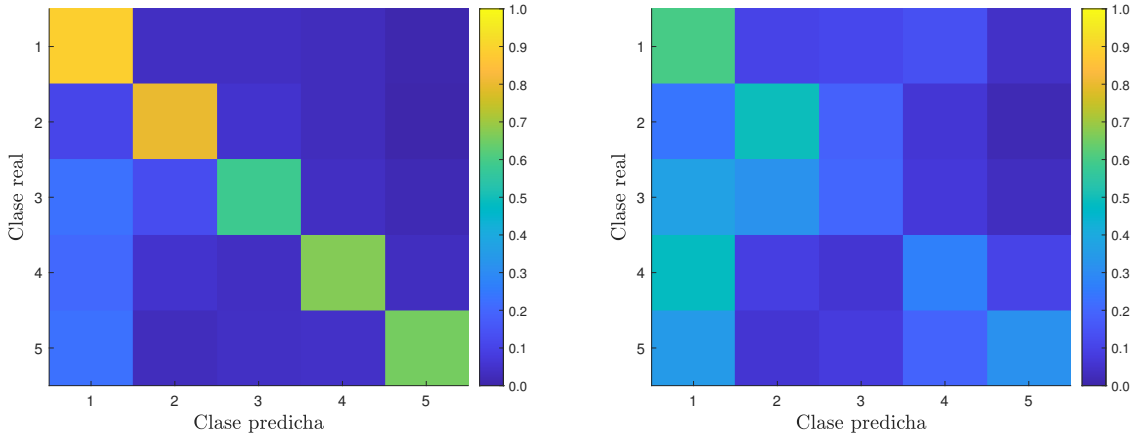
6.2.3 Matrices de confusión

Las matrices de confusión agregadas para la mejor arquitectura (i.e. 3.1.1) son mostradas en la figura 6.28 para los sets de entrenamiento y test.

Para el set de entrenamiento, la figura 6.28a muestra elevados ratios de clasificaciones correctas para todas las clases, siendo los más elevados para las clases 1 y 2. Además, se aprecia una ligera tendencia a confundir cualquier clase con la clase 1.

En cambio para el set de test, la figura 6.28b muestra un menor ratio de clasificaciones correctas para todas las clases y, una mayor tendencia a incurrir en la confusión de cualquier clase con la clase 1. Sin embargo, mantiene un ratio de clasificaciones correctas elevado para las clases 1 y 2 (i.e. un ratio de clasificaciones correctas de 0.60 y 0.49 para la clase 1 y 2, respectivamente), lo que explica que la barra de error asociada a la arquitectura 3.3.1 en la figura 6.27b se encuentre por encima del porcentaje de la clase más abundante.

Las matrices de confusión desagregadas son mostradas en la figura 6.29 para los sets de entrenamiento y test. Por un lado, la figura 6.29a muestra como la mayoría de espectros son clasificados correctamente en el set de entrenamiento. Sin embargo, varios espectros pertenecientes a la clase 3 son clasificados como clase 1. Estos espectros están formados por una semicircunferencia abierta por la aparición prematura del comportamiento lineal (i.e. el comportamiento lineal comienza a un valor de impedancia imaginaria negativa superior a lo común, lo que impide a la semicircunferencia cerrarse), como se muestra en la figura 6.30. Este tipo de patrón se repite de forma abundante en espectros de la clase 1, como se muestra en la figura 6.31. Por lo tanto, la red convolucional está captando esta combinación de patrones e identificándola como clase 1.



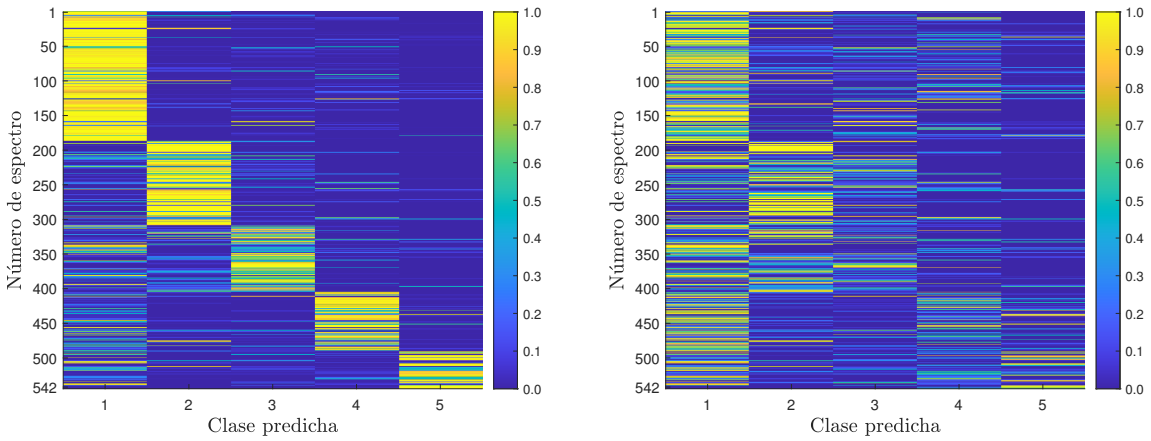
(a) Entrenamiento **(b)** Test

Figura 6.28: Matrices de confusión agregadas para la red convolucional optimizada (i.e. arquitectura 3.3.1). Cada matriz de confusión se ha construido a partir del entrenamiento (a) y testeo (b) de la red convolucional optimizada 100 veces. Por lo tanto, la magnitud representada en la escala de color es el porcentaje de veces en tanto por 1 que el algoritmo ha clasificado cierta clase como 1, 2, 3, 4 o 5.

Por otro lado, la figura 6.29b muestra como el ratio de acierto para el set de test es muy inferior. Sin embargo, se aprecian ciertos espectros pertenecientes a la clase 2 son clasificados de forma correcta. La mayoría de estos espectros están formados por dos semicircunferencias, como se muestra en la figura 6.32. Por lo tanto, la red convolucional está captando los patrones formados por dos semicircunferencias y, clasificándolos como clase 2.

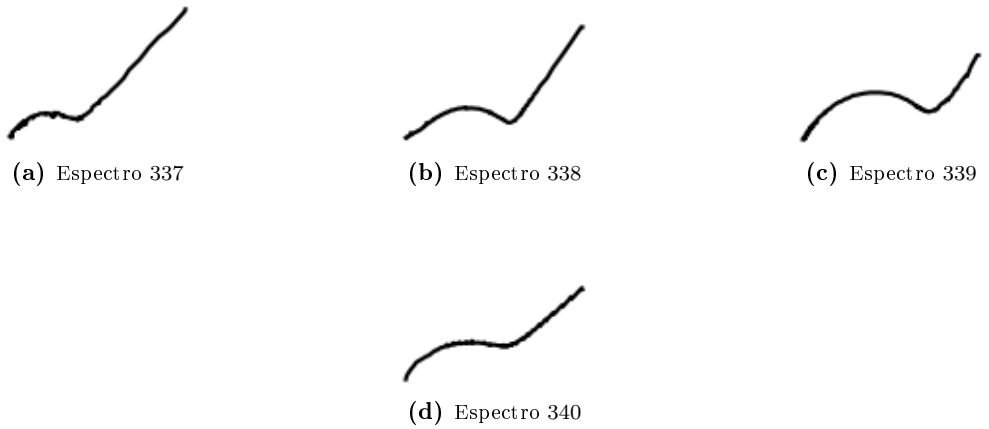
6.2.4 Tiempos de clasificación

La distribución del tiempo de clasificación (i.e. el tiempo transcurrido en clasificar un ejemplo por el algoritmo ya entrenado) es mostrado en la figura 6.33. El tiempo medio es 16.41 ms y,los valores que engloban al 95 % de la población son 16.02 ms y 16.95 ms.



(a) Entrenamiento **(b)** Entrenamiento

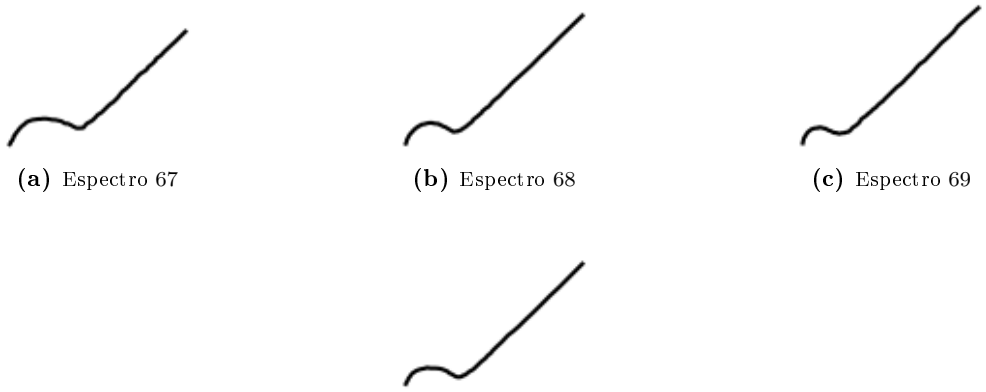
Figura 6.29: Matrices de confusión desagregadas para la red convolucional optimizada (i.e. arquitectura 3.3.1). Cada matriz de confusión se ha construido a partir del entrenamiento (a) y testeo (b) de la red convolucional optimizada 100 veces. Por lo tanto, la magnitud representada en la escala de color es el porcentaje de veces en tanto por 1 que el algoritmo ha clasificado cierto espectro como clase 1, 2, 3, 4 o 5.



(a) Espectro 337 **(b)** Espectro 338 **(c)** Espectro 339

(d) Espectro 340

Figura 6.30: Espectros pertenecientes a la clase 3 clasificados como clase 1 por la red convolucional con la arquitectura 3.3.1



(a) Espectro 67 **(b)** Espectro 68 **(c)** Espectro 69

(d) Espectro 70

Figura 6.31: Ejemplos de espectros pertenecientes a la clase 1 conformados por semicircunferencia abierta y una línea recta.

(a) Espectro 276 **(b)** Espectro 277

(c) Espectro 278 **(d)** Espectro 279

Figura 6.32: Espectros pertenecientes a la clase 2 conformados por dos semicircunferencias.

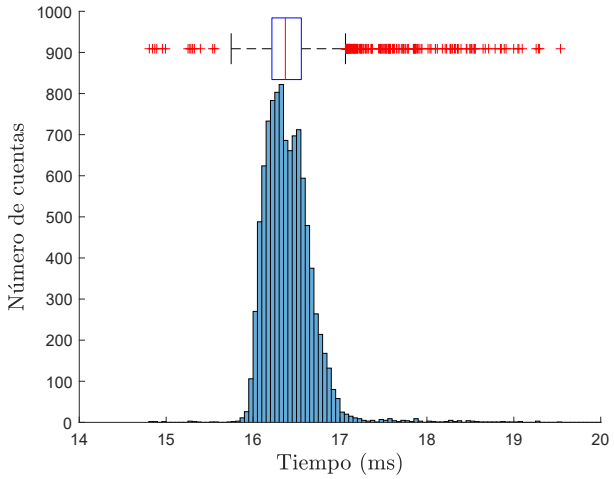


Figura 6.33: Distribución de tiempos de clasificación correspondientes a la red convolucional (arquitectura 3.3.1), para 10 000 repeticiones. La medida de tiempos de clasificación se ha realizado con un ordenador HP Pavilion Gaming Laptop 15-ec2xxx, con un procesador AMD Ryzen 5 con 12 CPUs y 3.3 GHz, una memoria RAM de 16 384 MB y una gráfica dedicada NVIDIA GeForce GTX 1650.

6.3 Selección del mejor algoritmo

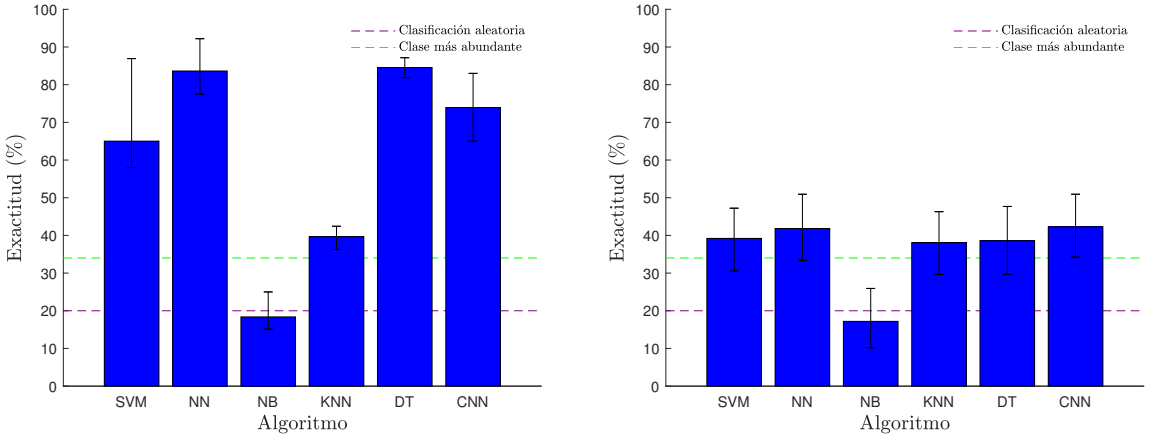
Una vez optimizados todos los algoritmos de Machine Learning y Deep Learning considerados en este Trabajo Final de Máster, se seleccionó el mejor algoritmo para la clasificación de espectros EIS. Para ello, se compararon las exactitudes obtenidas para cada algoritmo optimizado, como se muestra en la figura 6.34.

Los valores de exactitud para el set de entrenamiento se muestran en la figura 6.34a, donde se pueden observar unas exactitudes medias elevadas (i.e. por encima del 80 %) para los algoritmos Neural Network (NN) y Decision Tree (DT). Sin embargo, la exactitud que realmente marca la capacidad de clasificación es la del set de test, mostrada en la figura 6.34b, donde se puede observar que los algoritmos Neural Network (NN) y Convolutional Neural Network (CNN) presentan las exactitudes más elevadas con un 41.83 % y 42.27 %, respectivamente.

Este resultado es coherente con el problema planteado en este Trabajo Final de Máster (i.e. la clasificación de espectros EIS en base a sus diagramas de Nyquist), ya que, la mayor parte de la información en los diagramas de Nyquist no está contenida en los valores individuales de impedancia, sino que es la forma de diagrama lo que realmente aporta información. Por lo tanto, los algoritmos capaces de agregar la información de varios puntos y realizar la clasificación en base a dicha agregación, como los algortimos NN y CNN, tienen ventaja en este tipo de problemas.

Entre los algortimos NN y CNN no existe una diferencia estadísticamente significativa. Sin embargo, se selecciona el algoritmo CNN como la mejor opción por poseer un porcentaje medio de exactitud superior.

En cuanto a los tiempos de clasificación son comparados en la figura 6.35 donde, se puede observar unos tiempos de clasificación superiores para los algoritmos Convolutional Neural Network (CNN) y K-Nearest Neighbor (KNN).



(a) Entrenamiento **(b)** Test

Figura 6.34: Exactitud para diferentes algoritmos optimizados entre el set de entrenamiento (a) y test (b). Las barras de error para cada algoritmo han sido calculadas repitiendo cada entrenamiento 2000 veces y, eligiendo los valores que engloban al 95 % de la población obtenida. Además, se muestra la exactitud esperada para un clasificador aleatorio (morado) y, un clasificador cuya predicción sea siempre la clase más abundante (verde).

Por un lado, el algoritmo CNN cada vez que realiza una clasificación necesita pasar la imagen por diferentes etapas convolucionales y neuronales, lo que supone un elevado coste computacional y, por lo tanto, ralentiza el proceso de clasificación.

Por otro lado, el algoritmo KNN cada vez que realiza una clasificación necesita evaluar la distancia entre el espectro a clasificar con todos los espectros utilizados en el entrenamiento, lo que supone un tiempo de clasificación superior a otros algoritmos cuya clasificación es más sencilla (i.e. SVM, NN, NB y DT).

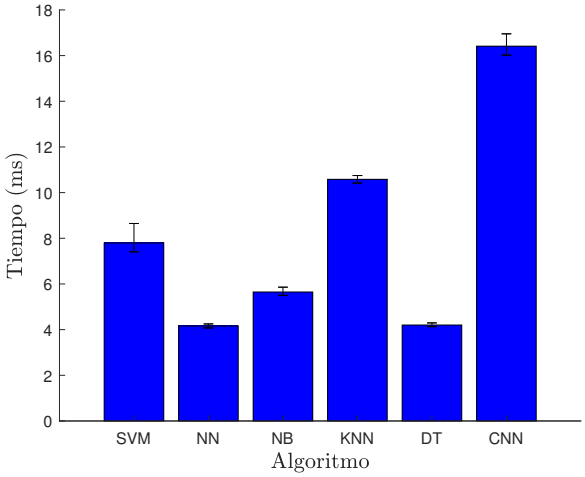


Figura 6.35: Distribución de tiempos de clasificación para diferentes algoritmos optimizados. Las barras de error se han obtenido mediante la medida de 10 000 tiempos de clasificación para cada algortimo y, eligiendo los valores que engloban al 95 % de la población. La medida de tiempos de clasificación se ha realizado con un ordenador HP Pavilion Gaming Laptop 15-ec2xxx, con un procesador AMD Ryzen 5 con 12 CPUs y 3.3 GHz, una memoria RAM de 16 384 MB y una gráfica dedicada NVIDIA GeForce GTX 1650.

6.4 Análisis de Componentes Principales

El análisis de componentes principales de componentes principales o PCA por sus siglas en inglés (Principal Component Analysis), cuyo resultado se muestra en la figura 6.36, permite obtener las siguientes conclusiones:

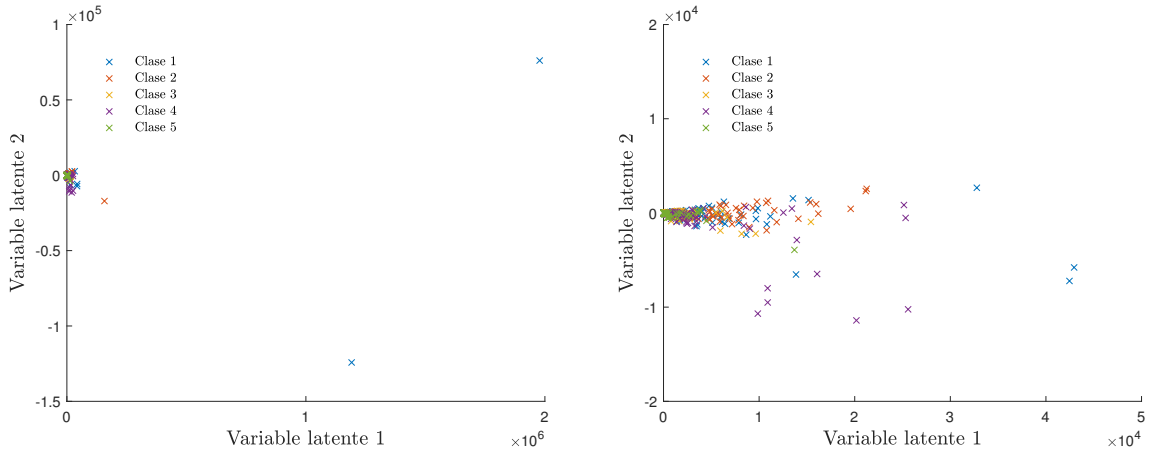
Es posible reducir las 160 características originales a 2 variables latentes y, aún así, explicar el 99.94 % de la variabilidad de espectros EIS almacenados en la base de datos refinada (el 99.53 % y 0.42 % la variable latente 1 y 2, respectivamente).

Además, permite identificar 3 espectros anómalos que se alejan de la tendencia general, ya que, poseen valores de impedancia inusualmente elevados (como se muestra en la figura 6.37). La presencia de espectros anómalos puede sesgar el análisis PCA y, por lo tanto, se repite eliminando las observaciones anómalas.

El análisis PCA sin datos anómalos, cuyo resultado se muestra en la figura 6.38, permite obtener las siguientes conclusiones:

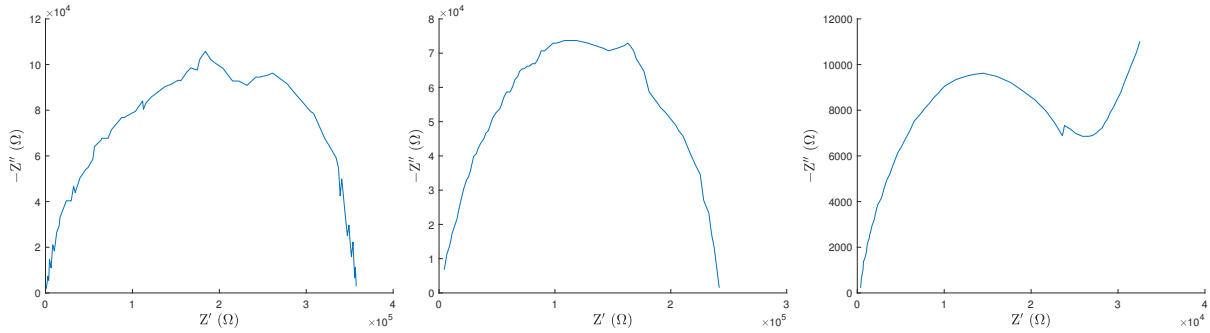
Es posible reducir las 160 características originales a 2 variables latentes y, aún así, explicar el 97.36 % de la variabilidad de espectros EIS almacenados en la base de datos refinada (el 88.34 % y 9.02 % la variable latente 1 y 2, respectivamente). Por lo tanto, los espectros anómalos presentaban una fuerte influencia sobre el primer análisis PCA, ya que, la variabilidad explicada por cada variable latente se ha modificado sustancialmente.

Además, se puede observar como las diferentes tipologías de circuitos eléctricos equivalentes consideradas están mezcladas entre sí sin un patrón que permita diferenciarlas, lo que explica la baja exactitud alcanzada en la secciones 6.1 y 6.2.



(a) Sin zoom **(b)** Zoom

Figura 6.36: Análisis PCA de todos los espectros EIS contenidos en la base de datos refinada.



(a) Espectro 134 **(b)** Espectro 135 **(c)** Espectro 207

Figura 6.37: Espectros EIS con variables latentes extremadamente diferentes del resto de espectros EIS.

Finalmente, se obtiene la importancia de las 160 características originales en la construcción de la variable latente 1 sin puntos anómalos, cuyo resultado se muestra en la figura 6.39, lo que permite obtener las siguientes conclusiones:

La importancia de las 80 primeras variables que pertenecen a valores de impedancia reales, es superior a las 80 restantes, que pertenecen al valores de impedancia imaginarios cambiados de signo. Esto se debe a que el rango de los valores reales ($-26.76\,\Omega$ - $9308.80\,\Omega$) es superior al rango de los valores imaginarios ($-59.49\,\Omega$ - $5511.5\,\Omega$) y, por lo tanto, los valores reales aportan más variabilidad a los espectros que los valores imaginarios.

Además se observa que, la importancia de la variable original es mayor cuanto más escorado a la derecha del diagrama de Nyquist se ha tomado el punto (i.e. a bajas frecuencias), lo que indica que la mayor parte de la variabilidad entre los espectros considerados está concentrada a bajas frecuencias.

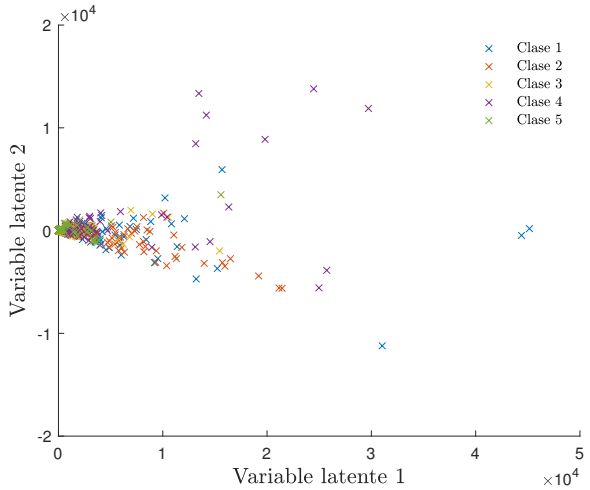


Figura 6.38: Análisis PCA de los espectros EIS (excepto los anómalos) contenidos en la base datos refinada.

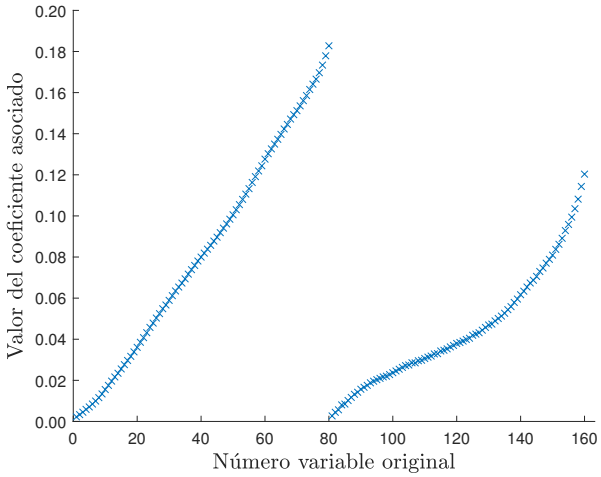


Figura 6.39: Identificación de las variables originales más importantes para la construcción de la variable latente 1 sin puntos anómalos. Donde, las primeras 80 variables corresponden con los valores reales de impedancia y, las 80 restantes con los valores imaginarios de impedancia cambiados de signo.

Capítulo 7

Conclusiones

7.1 Conclusiones

Los conclusiones derivadas de este Trabajo Final de Máster (TFM) son:

- Base de datos:
  - La construcción de la base de datos con diagramas de Nyquist que ignoran la frecuencia permite utilizar un gran número de espectros disponibles en la literatura científica. Sin embargo, impide que los datos tomados para cada espectro sean comparables, ya que, son tomados a frecuencias arbitrarias desconocidas, lo que dificulta su clasificación.
  - Las propuestas de circuitos eléctricos equivalentes en la literatura científica pueden no ser las más acertadas. Además, un mismo espectro EIS puede corresponderse con más de un circuito eléctrico equivalente, lo que dificulta la clasificación.
- Algoritmos de Machine Learning y Deep Learning:
  - La utilización de una base de datos desbalanceada (i.e. con distinto número de espectros para cada clase) propicia la confusión de las clases menos abundantes con las más abundantes.
  - El mejor algoritmo son las redes convolucionales con una exactitud media del 42.27 %. La superioridad de las redes convolucionales sobre otros algoritmos es debido a su capacidad de agrupar la información para formar patrones y, después realizar la clasificación sobre ellos.
- Análisis de componentes principales (PCA):
  - El análisis PCA permite obtener 2 variables latentes que explican el 97.36 % de la variabilidad de los espectros EIS contenidos en la base de datos. La representación de los espectros en base a las variables latentes no permite distinguir unas clases de otras, lo que concuerda con los bajos porcentajes de exactitud obtenidos.

- La importancia de cada variable original para explicar la variabilidad de los espectros aumenta al disminuir la frecuencia (i.e. cuanto más a la derecha en el diagrama de Nyquist aumenta la importancia). Por lo tanto, la información en los diagramas de Nyquist se concentra en la parte derecha de los mismos.

7.2 Trabajo futuro

Las posibles mejoras a los problemas identificados en este Trabajo Final de Máster (TFM) son:

- Generación de una base de datos sintética (i.e. una base de datos que contenga espectros EIS generados mediante simulación) que contenga información sobre los valores de impedancia y frecuencia de un elevado número de espectros EIS (i.e. 500 000 o más) y, mismo número de espectros EIS para cada clase.

  Esto permitiría realizar entrenamientos más completos, ya que, se tendría un mayor número de ejemplos y más completos (i.e. valores de impedancia y frecuencia). Además, los sesgos ocasionados por el número diferente de ejemplos en cada clase serían eliminados.

- Desarrollo de una metodología para cuantificar la similitud entre los espectros y así, poder distinguir entre clasificaciones erróneas evitables (i.e. ocasionadas por una confusión de la inteligencia artificial) y clasificaciones erróneas inevitables (i.e. ocasionadas por la similitud entre espectros EIS de las diferentes clases).

Bibliografía

Anyoha, Rockwell (2017). *The History of Artificial Intelligence*. Website. URL: `https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/`.

Åström, Karl Johan y Murray, Richard M (2021). *Feedback systems: an introduction for scientists and engineers*. Princeton university press.

Brunetto, Carmelo, Moschetto, Antonino y Tina, Giuseppe (2009). "PEM fuel cell testing by electrochemical impedance spectroscopy". En: *Electric Power Systems Research* 79.1, págs. 17-26.

Buchanan, Bruce G (2005). "A (very) brief history of artificial intelligence". En: *AI Magazine* 26.4, págs. 53-53.

Burkov, Andriy (2019). *The hundred-page machine learning book*. Vol. 1. Quebec City, Canada.

Chen, Di et al. (2016). "Deep multi-species embedding". En: *arXiv preprint arXiv:1609.09353*.

Cole, Kenneth S y Cole, Robert H (1941). "Dispersion and absorption in dielectrics I. Alternating current characteristics". En: *The Journal of Chemical Physics* 9.4, págs. 341-351.

Derr, Igor et al. (2016). "Degradation of all-vanadium redox flow batteries (VRFB) investigated by electrochemical impedance and X-ray photoelectron spectroscopy: Part 2 electrochemical degradation". En: *Journal of Power Sources* 325, págs. 351-359.

Dixit, Pooja y Prajapati, Ghanshyam I (2015). "Machine learning in bioinformatics: A novel approach for dna sequencing". En: *2015 fifth international conference on advanced computing & communication technologies*. IEEE, págs. 41-47.

Floridi, Luciano y Chiriatti, Massimo (2020). "GPT-3: Its nature, scope, limits, and consequences". En: *Minds and Machines* 30, págs. 681-694.

Fricke, Hugo (1932). "XXXIII. The theory of electrolytic polarization". En: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 14.90, págs. 310-318.

Fricke, Hugo y Morse, Sterne (1925). "The electric resistance and capacity of blood for frequencies between 800 and 41/2 million cycles". En: *The Journal of General Physiology* 9.2, pág. 153.

Frumkin, Alexander (1940). "Part II.(A) Electrokinetic equations. The study of the double layer at the metal-solution interface by electrokinetic and electrochemical methods". En: *Transactions of the Faraday Society* 35, págs. 117-127.

Goodfellow, Ian et al. (2016). *Deep Learning*. MIT Press. https://books. google. com/books.

Haenlein, Michael y Kaplan, Andreas (2019). "A brief history of artificial intelligence: On the past, present, and future of artificial intelligence". En: *California management review* 61.4, págs. 5-14.

Ioffe, Sergey y Szegedy, Christian (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". En: *International Conference on Machine Learning*. pmlr, págs. 448-456.

Jüttner, K (1990). "Electrochemical impedance spectroscopy (EIS) of corrosion processes on inhomogeneous surfaces". En: *Electrochimica Acta* 35.10, págs. 1501-1508.

Kok, Joost N et al. (2009). "Artificial intelligence: definition, trends, techniques, and cases". En: *Artificial Intelligence* 1, págs. 270-299.

Kowalczyk, Alexandre (2017). *Support vector machines succinctly*.

Krose, Ben y Smagt, Patrick van der (1996). *An introduction to neural networks*. The University of Amsterdam.

Lund, Brady D y Wang, Ting (2023). "Chatting about ChatGPT: How may AI and GPT impact academia and libraries?" En: *Library Hi Tech News*.

Lv, Jianfeng et al. (2023). "Diagnosis of PEM Fuel Cell System Based on Electrochemical Impedance Spectroscopy and Deep Learning Method". En: *IEEE Transactions on Industrial Electronics*.

Macdonald, Digby D (2006). "Reflections on the history of electrochemical impedance spectroscopy". En: *Electrochimica Acta* 51.8-9, págs. 1376-1388.

Meddings, Nina et al. (2020). "Application of electrochemical impedance spectroscopy to commercial Li-ion cells: A review". En: *Journal of Power Sources* 480, pág. 228742.

Memeti, Suejb y Pllana, Sabri (2018). "A machine learning approach for accelerating DNA sequence analysis". En: *The International Journal of High Performance Computing Applications* 32.3, págs. 363-379.

Nernst, Walther (1894). "Methode zur bestimmung von dielektrizitätskonstanten". En: *Zeitschrift für Physikalische Chemie* 14.1, págs. 622-663.

Newman, John y Balsara, Nitash P (2021). *Electrochemical systems*. John Wiley & Sons.

Orazem, Mark E y Tribollet, Bernard (2008). "Electrochemical impedance spectroscopy". En: *New Jersey* 1, págs. 383-389.

Poghossian, Arshak y Schöning, Michael J (2020). "Capacitive field-effect EIS chemical sensors and biosensors: A status report". En: *Sensors* 20.19, pág. 5639.

Randles, John Edward Brough (1947). "Kinetics of rapid electrode reactions". En: *Discussions of the faraday society* 1, págs. 11-19.

Steinbach, Michael y Tan, Pang-Ning (2009). "kNN: k-nearest neighbors". En: *The top ten algorithms in data mining*. Chapman y Hall/CRC, págs. 165-176.

Tabak, Michael A et al. (2019). "Machine learning to classify animal species in camera trap images: Applications in ecology". En: *Methods in Ecology and Evolution* 10.4, págs. 585-590.

Tingiris, Steve y Kinsella, Bret (2021). *Exploring GPT-3*. Packt Publishing.

Vidaković-Koch, Tanja et al. (2013). "Application of electrochemical impedance spectroscopy for studying of enzyme kinetics". En: *Electrochimica Acta* 110, págs. 94-104.

Xu, Ying et al. (2020). "Electrochemical impedance spectroscopic detection of E. coli with machine learning". En: *Journal of the Electrochemical Society* 167.4, pág. 047508.

Zhu, Shan et al. (2019). "Equivalent circuit model recognition of electrochemical impedance spectroscopy via machine learning". En: *Journal of Electroanalytical Chemistry* 855, pág. 113627.

Parte II

Presupuesto

Capítulo 1

Presupuesto desarrollo algoritmo de recomendación

Para el desarrollo del algoritmo de recomendación se han utilizado los recursos materiales, recursos inmateriales (i.e. software en este caso) y humanos, enunciados a continuación.

Los recursos materiales utilizados son un ordenador portátil modelo HP Pavilion Gaming Laptop 15-ec2xxx con procesador AMD Ryzen 5 5600H, memoria RAM de 16 384 MB y tarjeta gráfica dedicada NVIDIA GeForce GTX 1650, cuyo precio asciende a 920 € y, posee una vida útil de 4 años.

Los recursos inmateriales utilizados son el software Matlab®, cuyo precio más el de las toolboxes utilizadas (todo ello adquirido de forma anual) asciende a 2220 €.

Con los precios de los recursos materiales e inmateriales y, suponiendo 288 días laborables al año, con 8 h horas laborables cada uno, se obtiene el coste total del dichos recursos prorrateando de forma equitativa entre sus horas de uso, como se muestra en la tabla 1.1.

Tabla 1.1: Recursos materiales e inmateriales.

Concepto	Cantidad (h)	Coste unitario $\left(\frac{€}{h}\right)$	Coste (€)
Ordenador	300	0.027	8.10
Matlab®	300	1.146	343.80
Total			351.90

Los recursos humanos utilizados constan del tutor (catedrático de universidad a tiempo completo), co-tutor (investigador postdoctoral Juan de la Cierva a tiempo completo) y alumno (persona en prácticas a tiempo completo), cuyos salarios anuales ascienden a 47 114.16 € (según las tablas retributivas de la Universidad Politécnica de Valencia (UPV)), 30 000 € (según la Agencia Estatal de Investigación (AEI) para la convocatoria del año 2022) y 9907.20 € (según el Servicio Integrado de Empleo (SIE) de la UPV), respectivamente.

Con los precios de los recursos personales y, volviendo a suponer 288 días laborables al año, con 8 h horas laborables cada uno, se obtiene el coste total del dichos recursos prorrateando de forma equitativa entre sus horas de uso, como se muestra en la tabla 1.2.

Tabla 1.2: Recursos humanos.

Concepto	Cantidad (h)	Coste unitario $\left(\frac{€}{h}\right)$	Coste (€)
Tutor	100	20.449	2044.90
Co-Tutor	50	13.020	651.00
Alumno	300	4.300	1290.00
Total			3985.90

Con el coste de recursos materiales, inmateriales y humanos, se obtiene el presupuesto de ejecución material, que se muestra en la tabla 1.3.

Tabla 1.3: Presupuesto de ejecución material.

Concepto	Precio (€)
Recursos materiales e inmateriales	351.90
Recursos humanos	3985.90
Presupuesto de ejecución material	4337.80

El presupuesto de ejecución material por el desarrollo del algoritmo de recomendación asciende a: **cuatro mil trescientos treinta y siete euros con ochenta céntimos.**

Finalmente, se obtiene el presupuesto base de licitación considerando los gastos generales y el I.V.A, que se muestra en la tabla 1.4.

Tabla 1.4: Presupuesto base de licitación.

Concepto	Precio (€)
Presupuesto de ejecución material	4337.80
Gastos generales (13 %)	563.81
Presupuesto ejecución por contrata	4901.61
I.V.A (21 %)	1029.34
Presupuesto base de licitación	5930.95

El presupuesto de ejecución material por el desarrollo de la inteligencia artificial asciende a: **cinco mil novecientos treinta euros con noventa y cinco céntimos.**