ULPGC
Universidad de
Las Palmas de
Gran Canaria

Escuela de
Ingeniería Informática

Detección del deterioro cognitivo leve. Una propuesta basada en redes neuronales artificiales híbridas

Trabajo de fin de grado en Ingeniería Informática

Graciela Sabina Sánchez

Tutores:
Carmen Paz Suárez Araujo

Ylermi Cabrera León

07/2021

Agradecimientos

Gracias a mis tutores, Carmen Paz Suárez e Ylermi Cabrera, por ofrecerme esta propuesta en unos campos tan apasionantes como son el Alzheimer y la inteligencia computacional, así como por su tiempo y dedicación en este proyecto a pesar de las condiciones que hemos vivido este año.

Quiero hacer especial mención en agradecimiento también a Patricio García Báez de la Universidad de La Laguna por aportarme un código valioso que me ha facilitado la implementación del proyecto, y a Ludmila I. Kuncheva de Bangor University por la gran explicación que me ayudó a entender los métodos para construir las curvas ROC cuando se trabaja con ensembles.

Finalmente, agradezco a mis padres por haber estado apoyándome todo el tiempo para seguir adelante y a todos los amigos que estuvieron ahí para mí, sin el ánimo de los cuales no habría llegado hasta aquí.

Contenido

Índice de figuras

Índice de tablas

Capítulo 1: Introducción

En este capítulo se hablará del marco de este trabajo de fin de título. Primero se hablará de la motivación, los objetivos y la metodología y herramientas utilizadas para llevarlo a cabo. Luego se describirá el ámbito tratado en este trabajo, siendo estos puntos el estado del arte y los términos relevantes.

Motivación

La demencia se define como el "deterioro progresivo de las facultades mentales que causa graves trastornos de conducta" (Real Academia Española s. f. definición 2). Se trata, en esencia, de una pérdida de la función cerebral que puede afectar a la memoria, el lenguaje o el comportamiento, y cuyos efectos son irreversibles mayoritariamente. Esta enfermedad afecta a la población de edad avanzada y la probabilidad de padecer alguno de sus tipos aumenta a medida que la persona envejece (Alzheimer's Association 2021).

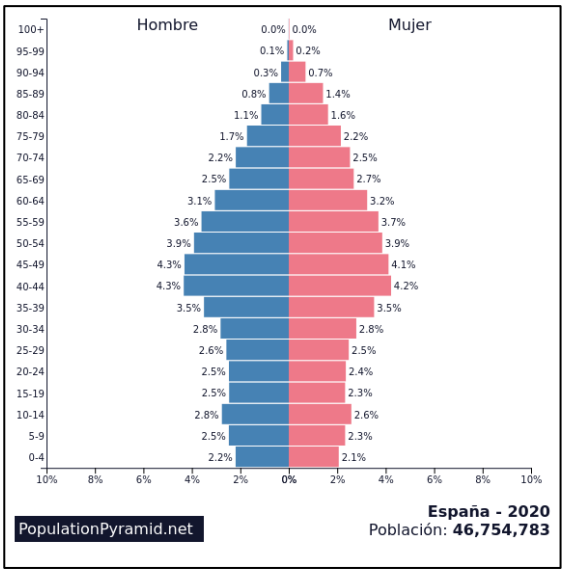

Figura 2. Pirámide que representa el envejecimiento de la población española y su posible impacto en el futuro. Fuente: populationpyramid.net

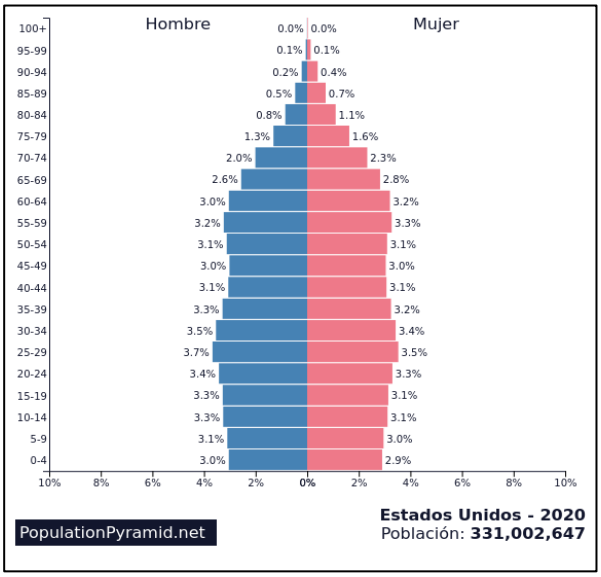

Figura 1. Pirámide que representa el envejecimiento de la población estadounidense y su posible impacto en el futuro. Fuente: populationpyramid.net

La Enfermedad de Alzheimer (EA) es el tipo de demencia más común y se pronostica que afectará cada vez a más personas al haber un envejecimiento alto de la población (Figura 2 y Figura 1). Debido a que es una enfermedad neurodegenerativa con graves consecuencias socioeconómicas, la detección temprana es un tema de gran importancia.

Existe un estado previo a la demencia llamado Deterioro Cognitivo Leve (DCL), que en la mayoría de los casos converge en la EA. En ese estado los síntomas no son muy evidentes, y su diagnóstico es complicado. Por ello, se han volcado diversos científicos desde diferentes ámbitos en la búsqueda de un método que lo facilite.

Los métodos de detección computacionales, concretamente la computación inteligente, pretenden mejorar la detección temprana de la EA, así como determinar con más exactitud el tratamiento adecuado y moderar el avance del deterioro cognitivo sufrido por la persona.

Objetivos

Al tratarse de un problema cuya complejidad puede escalar de manera sencilla, el estudio estará centrado en un paradigma binario, es decir, si el paciente tiene o no tiene Deterioro Cognitivo Leve (DCL).

Los objetivos que se proponen en este trabajo para presentar una solución de detección precoz de la Enfermedad de Alzheimer son:

- Análisis de la idoneidad de las Redes Neuronales Artificiales Híbridas para el diagnóstico temprano de la Enfermedad de Alzheimer.
- Determinar las mejores combinaciones de criterios clínicos para la detección del Deterioro Cognitivo Leve.
- Comparación del rendimiento de un ensemble neuronal y una Red Neuronal Artificial Híbrida.

Metodología y herramientas

Para desarrollar este proyecto y alcanzar los objetivos propuestos, se ha seguido una metodología de desarrollo software de tipo espiral, representada en la Figura 3 con sus cuatro fases, que permite centrarse en objetivos concretos y corregir el avance con cada iteración. De esta manera, se ha dividido el problema en distintas partes, que pretenden facilitar la verificación de los resultados en un corto periodo.

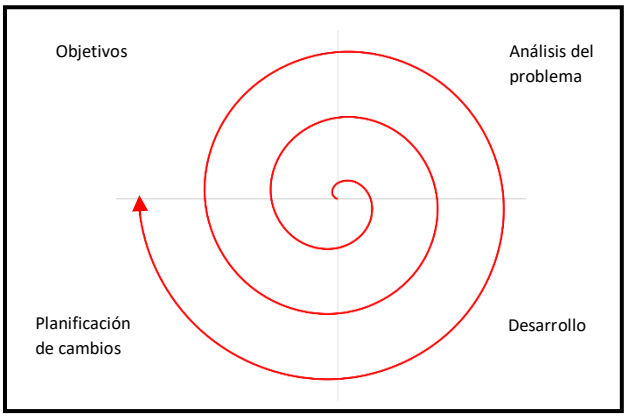

Figura 3. Metodología de tipo espiral.

Las tareas que se han llevado a cabo han seguido el siguiente orden:

- Estudio del campo del Deterioro Cognitivo Leve y la Enfermedad de Alzheimer: terminología, criterios clínicos, datos, etc.
- Estudio de las Arquitecturas Neuronales, en especial las Híbridas y los Ensembles Neuronales.
- Analizar investigaciones previas de sistemas inteligentes aplicados al campo de estudio.
- Desarrollar un sistema de ayuda al diagnóstico basado en Redes Neuronales Híbridas.
- Desarrollar un sistema de ayuda al diagnóstico basado en Ensembles Neuronales.
- Comparación de resultados con estudios previos y entre los sistemas desarrollados.

Para la realización de estas tareas, se han utilizado diversas herramientas que han facilitado enormemente la implementación del sistema de detección. Dado que las tareas prácticas de este trabajo se limitan al tratamiento de datos y la creación de los sistemas inteligentes, cada herramienta estará especializada en unas tareas concretas.

La primera herramienta utilizada ha sido Python 3.7.7 (Python Software Foundation 2021) por medio del manejador de paquetes *open source* mini-Conda. Junto a este se ha instalado el entorno de desarrollo Spyder (Spyder IDE 2020) para el desarrollo de los *scripts* en Python. Esta herramienta ha sido utilizada para un análisis de los datos inicial en forma de estadística descriptiva, que se completa posteriormente con la siguiente herramienta. Pandas, *Numpy* y *MatplotLib* son paquetes que también se usaron para desarrollar esta parte.

MatLab R2020a (The MathWorks, Inc. 2020) ha sido la herramienta principal con la que se ha desarrollado el trabajo. Se trata de una plataforma de programación y cálculo numérico que permite el análisis y tratamiento de datos, y la programación de algoritmos y modelos computacionales. Ofrece un lenguaje de *scripting* sencillo de aprender y manejar, pero muy potente para la creación de aplicaciones intensas en cálculo.

Dentro de MatLab se ha utilizado la Kohonen and CPANN Toolbox, creada por del grupo de investigación Milano Chemometrics y QSAR (Ballabio, Consonni, and Todeschini 2009; Ballabio and Vasighi 2012). Esta Toolbox proporciona una implementación verificada de las Redes Neuronales Artificiales de mapas autoorganizativos Kohonen y Counterpropagation, y se encuentra bajo la licencia *Creative Commons* para su uso y distribución libre.

Justificación de las competencias específicas cubiertas

Tabla 1. Tabla de competencias y justificaciones cubiertas.

Código	Descripción	Justificación
IS03	Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles	Aplicación de técnicas del software para el desarrollo de un programa que permite usar de manera transparente un módulo de inteligencia artificial construido en Matlab
IS04	Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales	Diseño de un subprograma que conforma el *Wrapper* de atributos para la alimentación automatizada de datos al sistema inteligente
CP05	Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano en una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes o entornos inteligentes	Aplicación de técnicas estudiadas para el tratamiento de los datos para el sistema inteligente desarrollado
CP07	Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos	Desarrollo de un ensemble neuronal, que aúna las decisiones de distintos sistemas inteligentes y toma su propia decisión final a partir de las anteriores por medio de técnicas computacionales

Aportaciones al entorno socioeconómico

El desarrollo de este trabajo trata de aportar una herramienta para la detección temprana del deterioro cognitivo leve (DCL), lo cual implica:

- Eficacia de los tratamientos. Cuanto antes se comiencen las terapias, más posibilidades habrá de controlar y retrasar el avance de la enfermedad, mejorando su calidad de vida (Sara López Álava 2021).
- Adaptación. Conociendo la enfermedad antes de que sea demasiado evidente, tanto el paciente como la familia podrán prepararse para hacerle frente al avance (Know Alzheimer - Neurología 2014).
- Toma de decisiones. El paciente tendrá las facultades necesarias para tomar decisiones, permitiendo ejercer su derecho sobre su propia salud (Equipo docente Grupo Sinapsis 2019).
- Mayor seguridad. Podría evitar situaciones peligrosas para el paciente como caídas, abusos domésticos o económicos, problemas de salud derivados de la pérdida de movilidad, entre otros.
- Reducción de costes. "En 2010, el gasto mundial total estimado en demencia fue de US$ 604.000 millones" (Mónica Aranda and Alejando Calabria 2019). Estos gastos incluyen consultas, medicamentos y estudios complementarios, que pueden preverse y dar el mejor tratamiento posible al paciente.

Estado del arte

A medida que la demencia y la Enfermedad de Alzheimer (EA) han ido cobrando importancia en la actualidad, junto con los avances tecnológicos se han empezado a utilizar métodos computacionales y de inteligencia artificial con el objetivo de aportar una nueva visión en las numerosas investigaciones. Son diversos los estudios que buscan mejorar la atención a los pacientes con esta enfermedad, haciendo uso de los medios actuales por los profesionales médicos para facilitar los procesos y aportar una visualización más precisa de las exploraciones realizadas a los pacientes (Carlos Bonilla 2019).

Actualmente, un gran número de estudios aplican las redes neuronales artificiales (RNA) para la detección de estas enfermedades, variando ampliamente los tipos de redes y de datos utilizados, que tienen en cuenta una mayor cantidad de elementos para la toma de decisiones, permitiendo hacer aproximaciones fiables. Por lo general, los sistemas de aprendizaje profundo pueden diagnosticar con un 90% de acierto a los enfermos de EA, mientras que en los casos de detección precoz desciende ligeramente a un porcentaje superior al 80% (STIMULUS SALUD, S.A. 2017).

Se ha extraído una muestra variada de estudios en la Tabla 2 que utilizan RNAs, y que servirán de base inicial para el enfoque en sistemas neuronales de este proyecto. De entre estos, se ha notado una tendencia hacia el uso del aprendizaje profundo (AP, del inglés *Deep learning)* con un amplio rango de tipos de datos, ya que es un método que permite mejorar la precisión ante relaciones de alta complejidad (Stamate et al. 2020).

También se han encontrado RNAs más comunes a modo de experimento con tipos de datos muy específicos, como ha sido el caso de las redes convolucionales o las redes *backpropagation*. Suelen aportar ejemplos de detección con el objetivo de la comparación con

métodos no neuronales, o la comprobación del beneficio al usar pruebas poco comunes (Mukhtar and Farhan 2020).

Y combinando varios clasificadores, muchas veces sesgados por sus decisiones, se encuentran los ensembles neuronales (Báez et al. 2012). Aunque en un porcentaje bastante más reducido, estos últimos aportan una solución cuyo objetivo sea mejorar la detección con modelos que no tienen que ser necesariamente buenos.

La mayor parte de los estudios hacen uso de conjuntos de datos procedentes de ADNI, puesto que es una de las bases de datos de Alzheimer más conocidas (Michael W. Weiner 2017). De ella se pueden obtener datos de neuroimagen, tests neuropsicológicos, demográficos y biomarcadores, lo que la convierte en una de las mejores fuentes para los proyectos de ayuda a la detección de la EA. Por el otro lado, se suele hacer uso de pruebas confeccionadas especialmente para el estudio, con preguntas centradas en el problemas a abordar por la investigación (Wang et al. 2019).

A pesar de la gran cantidad de datos manejados y los métodos de RNAs con mayor y menor complejidad, la mayoría de los estudios se centran en problemas binarios, dado que en problemas de tres clases o más la complejidad aumenta enormemente y los resultados obtenidos puede que no siempre sean los esperados (Cabrera-Leon et al. 2018). Los tipos de pacientes suelen padecer de EA y los intentan diferenciar de otros tipos de estados, o pacientes con un tipo de demencia y algún tipo de dolencia extra que se considere un riesgo para el avance de la enfermedad (Mato-Abad et al. 2018).

De estos estudios se ha concluido que la mayor parte de las investigaciones actuales y pasadas utilizan imágenes de resonancia magnética (MRI) y especímenes biológicos para trabajar (Liu et al. 2018). Además, los métodos computacionales más comunes son las Máquinas de Soporte de Vectores y las redes neuronales. Estas últimas se utilizan de manera general, muchas veces para comparativas con otros métodos, y en algunas ocasiones se usan algunos tipos concretos como la red Counterpropagation (CPN) o el Deep Learning.

Tabla 2. Trabajos previos de detección de demencia con métodos de computación inteligente.

Dataset	Método	Tipo de pacientes	Tipos de datos	Resultados	Referencia
Custom test from NDUHCSC[1]	MLP[2]	MCI con depresión y sin depresión	GDS[3] y otros tests neuropsicológicos	Acc[4]: 86%	(Mato-Abad et al. 2018)
ADNI	MLP y ConvBLSTM[5]	CN, MCI y AD[6]	Demográficos, tests neuropsicológicos, MRI[7], PET[8] y CSF[9]	Acc: 86%	(Stamate et al. 2020)
ADNI	CNN[10]	MCIc, MCInc[11]	MRI, APOE[12], MMSE[13] y CDR[14]	Acc: 94%	(Mukhtar and Farhan 2020)
ADNI	CPN[15]	CN, EMCI y LMCI[16]	NPI, MoCA, FAQ [17]y demográficos	Acc: 54%	(Cabrera-Leon et al. 2018)
Custom surveys	BP [18]	CN, AD	MMSE, ADL[19] y biomarcadores	Acc: 92%	(Wang et al. 2019)
Custom surveys	SVM, BP y RF[20]	MCI y AD	Tests neuropsicológicos y demográficos	Acc: 88%	(Guo et al. 2021)
ADNI	Ensemble de HUMANN-S	AD, VD[21]	MMSE, FAQ, Katz's Index, Barthel's Index y Lawton-Brody's index	Error: ±20%	(Báez et al. 2012)

[1] Neurology Department of the University Hospital Complex of Santiago de Compostela.
[2] Multi-Layer Perceptron.
[3] Geriatric Depression Scale.
[4] Accuracy.
[5] Convolutional Bidirectional Long Short-Term Memory.
[6] Alzheimer Disease.
[7] Magnetic Resonance Imaging.
[8] Positron Emission Tomography.
[9] Cerebrospinal Fluid.
[10] Convolutional Neural Network.
[11] MCI convertors y MCI nonconvertors.
[12] Apolipoprotein E.
[13] Mini-mental State.
[14] Clinical Dementia Rating.
[15] Counterpropagation network.
[16] Control (sanos), Early Mild Cognitive Impairment y Late Mild Cognitive Impairment.
[17] Neuropsychiatric Inventory, Montreal Cognitive Assessment y Functional Activities Questionnaire.
[18] Backpropagation.
[19] Activities of Daily Living.
[20] Support Vector Machine y Random Forest.
[21] Vascular Dementia.

Criterios diagnósticos en la enfermedad de Alzheimer

Actualmente, no existe un método infalible para la detección del deterioro cognitivo ni de la Enfermedad de Alzheimer (EA), con lo cual esta evaluación se somete a criterio clínico. En el Protocolo de Diagnóstico del Deterioro Cognitivo en la Comunidad Autónoma de Canarias (Gobierno de Canarias 2019), se establece una serie de pautas para que los profesionales puedan llevar a cabo esta evaluación de la manera más eficaz posible, clasificando las pruebas de detección en los siguientes grupos:

- **Anamnesis[22] cognitiva global**: Comprende los aspectos cognitivos, funcionales y psicopatológicos y conductuales, así como factores y hábitos de riesgo, nivel educacional, enfermedades genéticas y medicamentos con efectos que puedan afectar al área cognitiva.
- **Exploración física y neurológica básica:** Permiten indagar en causas secundarias del deterioro cognitivo que puedan tener tratamientos específicos.
- **Test cognitivos breves**: Utilizados para una valoración mental estructurada, cuantificando el deterioro. Los resultados se encuentran influidos por la edad, el déficit sensorial, el nivel cultural y el idioma de la persona por lo que son un complemento al estudio.
- **Análisis clínico**: Pruebas complementarias de laboratorio que permiten descartar causas potencialmente reversibles o la presencia de otras enfermedades que contribuyan al deterioro cognitivo.
- **Neuroimagen**: Valoración del grado y distribución de la atrofia cerebral, así como la presencia de posibles lesiones vasculares.
- **Biomarcadores**: Tienen por objetivo afinar el diagnóstico etiológico, realizándose solo a nivel especializado en casos determinados.

El proceso de evaluación del deterioro cognitivo se lleva a cabo a lo largo de un plazo de tiempo, que permite el análisis de su evolución para determinar si es un avance muy rápido y está causado por otros factores.

Bases de datos de demencia

Existen varias bases de datos de pacientes con demencia disponibles para la investigación. Dado que se trata de datos de carácter personal, muchas de ellas ofrecen los datos por medio de una petición formal que garantice su uso adecuado.

En la Tabla 3 se encuentra la información de las bases de datos encontradas durante la realización del proyecto. Debido al volumen de criterios utilizados para detectar el deterioro cognitivo y su grado, todas las bases de datos ofrecen conjuntos con un número de pacientes variable, clasificados por los tipos de pruebas que se les ha realizado.

De las bases de datos encontradas, solo tres de ellas son de acceso privado (Dementia TalkBank, NIAGADS y Sidney memory and ageing study), aunque todas ofrecen un contacto para el acceso a los datos. Por la parte de las públicas, lo normal es la visualización o descripción de

[22] Una anamnesis se define como la "información aportada por el paciente y por otros testimonios para confeccionar su historial médico" (Real Academia Española 2021b). Se trata del paso previo en la examinación clínica, compuesto por un interrogatorio que permite conocer las dolencias del paciente y su situación.

forma no restringida, y la descarga de los datos por medio de una petición con el fin de garantizar la protección de datos de los pacientes.

Para un mejor análisis, el número de muestras es sumamente importante. Aquellas con el mayor número de tipos de datos y mayor número de muestras en ellos son los más deseables, pues permiten una mejor generalización cuando se aplican a estudios con métodos de inteligencia artificial.

En cuanto a tipo de datos, aquellas que ofrecen el mayor número (ADNI, DPUK, Tadpole y Sidney MAS) resultan ser las más interesantes. De esta forma, se tiene un mayor conjunto de clasificaciones para comparar. Adicionalmente, Dementia TalkBank ofrece datos de distinto origen, lo que podría llevar a un estudio más específico del diagnóstico de la demencia.

Analizando el número de muestras, las más visibles a la comunidad parecen ser las que cuentan con las mejores características en este sentido. Esto se aprecia en ADNI u OASIS, siendo ambas de hace más de 10 años y con actualizaciones de los datos a lo largo de este periodo.

Atendiendo a las características de cada una y las observaciones individuales, así como su acceso, las que ofrecen mejores conjuntos de datos son: ADNI, OASIS, DPUK y Rdatasets.

Tabla 3. Bases de datos y repositorios de demencia.

Nombre	Tipos de datos	Número de muestras	Acceso	Referencias
ADNI	Clínicos, genéticos, MRI, PET y Bioespecímenes	1070-2000 (ADNI-3)	Público con autorización	(Michael W. Weiner 2017)
Dementia TalkBank	Vídeo y audio transcritos	Variable según dataset	Privado	(DementiaBank consortium group 2017)
OASIS Brains Datasets	MRI y PET	MRI: 2168 PET: 1608 TOTAL: 1098	Público con autorización	(XNAT 2007)
NHS Digital	Clínicos	15000	Público	(NHS Digital 2018)
NIAGADS	Genotipos, secuencias y expresiones genéticas	Total: 34236 Genotipos: 24000547256	Privado; posible acceso para investigación	(NIA 2016)
NCBI	Genotipos y expresiones genéticas	Variable según dataset	Público	(National Center For Biotechnology Information 2016)
PacBIO	Genotipos y expresiones genéticas	21742	Público	(Pacific Biosciences of California, Inc. 2016)
Dryad Digital Repository	Cognitivo y PET	Variable según dataset	Público	(DRYAD 2009)
DPUK Data Portal	Clínicos, genéticos, MRI, PET y bioespecímenes	Variable según dataset	Público con autorización	(Dementias Platform UK 2020)

Redes neuronales artificiales

Las Redes Neuronales Artificiales (RNA) son un modelo computacional inspirado en las Redes Neuronales Biológicas (RNB) del sistema nervioso, centrado concretamente en el cerebro humano. De las RNB se obtienen los principales conceptos e ideas, que luego se modelan con el objetivo de encontrar soluciones a problemas de otros campos (Freeman and Skapura 1991).

"Una neurona es un tipo de célula altamente especializada, que compone el sistema nervioso" (María Estela Raffino 2020). Está compuesta de tres partes principales: las dendritas, que reciben los impulsos de información de otras neuronas, el soma es donde los procesa, para luego transmitirlos por el axón hacia otras. Modelando a partir de esta estructura básica, se forma la unidad mínima de una RNA: la neurona artificial o nodo (Figura 4).

Una neurona artificial también cuenta con unas entradas de información, que están asociadas con una cantidad denominada peso, y una única salida (Freeman and Skapura 1991). Las salidas pueden convertirse en conexiones, que permiten unir a las neuronas artificiales entre sí de manera directa o indirecta para propagar el valor a las siguientes neuronas. Esta última conexión forma una RNA.

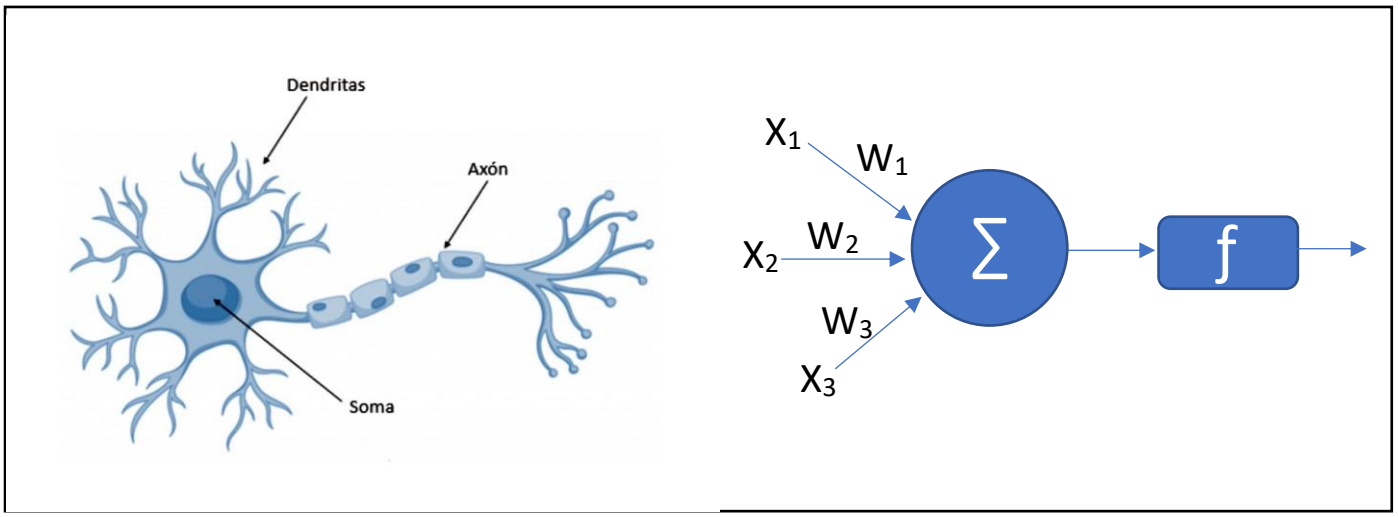

Figura 4. Comparación entre una neurona biológica (Fuente*: García-Olalla Olivera, s. f.*) y el modelo de una neurona artificial.

Existen diversas formas de agrupar las redes neuronales para distinguirlas por sus características. La primera asociación se puede hacer según el tipo de conexiones que tenga una RNA, lo que las clasifica en dos grupos (A. K. Jain, Jianchang Mao and K. M. Mohiuddin 1996):

- *Feed-forward*: La red no cuenta con conexiones que produzcan alguna clase de bucle. "En general, las redes *feed-forward* son estáticas, es decir, producen solamente un conjunto de valores de salida en vez de una secuencia de valores para una entrada dada" (A. K. Jain, Jianchang Mao and K. M. Mohiuddin 1996).
- *Recurrent* o *feedback:* Existen bucles de conexiones que regresan a nodos anteriores. Cada vez que recibe una entrada de datos, calcula la salida y modifica las entradas por medio de las conexiones hacia atrás.

En última instancia, las redes neuronales artificiales son un modelo matemático que pretende "aprender" a partir de unas condiciones concretas que se le dan, y crear una solución general. Esto implica que son altamente dependientes del tipo de problema en el que se

apliquen. Atendiendo a esta característica, existe otro modo de agrupar a las redes en tres tipos de aprendizajes distintos (A. K. Jain, Jianchang Mao and K. M. Mohiuddin 1996):

- Aprendizaje supervisado: A partir de una serie de datos de ejemplo y sus resultados, se busca una función de activación que permita deducir el valor de cualquier observación.
- Aprendizaje no supervisado: El modelo se ajusta a las observaciones, clasificando los datos según sus características.
- Aprendizaje híbrido: Combina ambos tipos anteriores, produciendo una parte de los pesos con aprendizaje supervisado y la otra con aprendizaje no supervisado.

Existe un otro tipo de aprendizaje no neuronal: el aprendizaje por refuerzo. En este tipo de aprendizaje, un agente recibe la información directamente del entorno en el que se encuentra, y aprende a reaccionar ante él recibiendo *feedback,* en forma de recompensas, del cambio que producen sus acciones. Así pues, se puede considerar una variante del aprendizaje supervisado.

Cada aprendizaje tiene su objetivo, siendo útil en casos distintos. Centrándonos en los dos primeros tipos de aprendizajes aplicables a las redes neuronales, el aprendizaje supervisado se caracteriza por necesitar etiquetados en los datos para ir perfeccionando la función activadora. Por el otro lado, el aprendizaje no supervisado es capaz de inferir los etiquetados de los datos, aunque necesita una validación final de los datos (Julianna Delua 2021).

Para un problema de clasificación binario, por tanto, sería posible utilizar tanto aprendizaje supervisado como no supervisado dependiendo de con qué información contemos.

Las reglas de aprendizaje de Grossberg

Stephen Grossberg es conocido por su extenso estudio en la creación de modelos matemáticos del cerebro humano para el reconocimiento visual, del habla, la memoria, la percepción cognitiva, así como del desarrollo y su organización. Entre sus numerosos estudios, definió dos reglas de aprendizaje, según la relación de las neuronas, denominadas: *Instar* y *Outstar*.

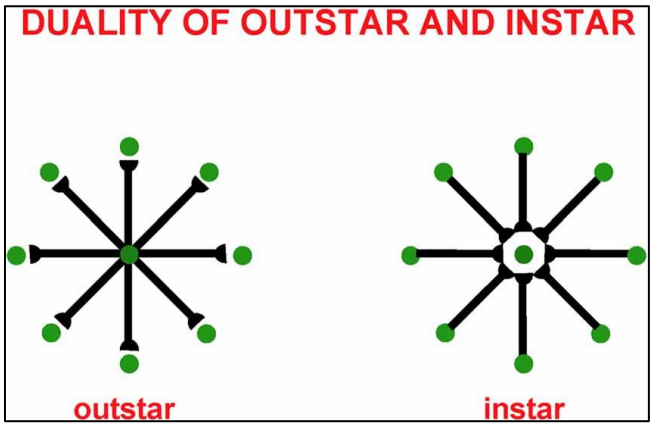

Figura 5. Representación de nodos Outstar e Instar
(Fuente: *Grossberg 2020)*

Una red Outstar se forma con una neurona central conectada a aquellas que la rodean. Cuando esta neurona aprende, envía una señal por todas las conexiones que modifica los pesos acorde a la información que va recibiendo con el paso del tiempo. Por otro lado, la red Instar se

caracteriza por invertir el sentido de las conexiones de la Outstar, como puede observarse en la Figura 5. En esta, cuando la neurona central aprende, recibe las señales de las neuronas que la rodean y modifica los pesos (Grossberg 2020).

En ambas, los pesos se van adaptando para formar un patrón espacial según la actividad de las neuronas, aumentando o disminuyendo si hay más o menos señales. Con esto, la red es capaz de aprender y recordar patrones arbitrarios a través de todo el conjunto de neuronas, por lo que son aplicables junto con redes dedicadas a la categorización de datos.

Mapas auto-organizativos (SOM) o Red de Kohonen

La Red de Kohonen, también denominada Mapa Auto-Organizativo (SOM, del inglés *Self-Organizing Map*), es un tipo de red neuronal artificial creada para la clusterización de datos en grupos por características. Fue propuesta por Teuvo Kohonen en 1982, tras la observación de cómo el córtex cerebral reagrupaba su estructura dependiendo del tipo de actividad que se pudiera ejercer (Freeman and Skapura 1991).

Esta red hace uso de un aprendizaje no supervisado. Esta red activa las neuronas de salida por competición de manera que, del conjunto de neuronas disponibles en la capa de salida, se activará aquella cuyo valor supere al de las de su alrededor. Cuando haya entrenado lo suficiente, se habrá producido una representación discreta del espacio de muestras, también denominado mapa.

Debido a que la red crea las agrupaciones a partir de las características que detecte durante el entrenamiento, el mayor inconveniente de esta red es que siempre realizará el procedimiento de aprendizaje completo cuando recibe datos nuevos.

La arquitectura este tipo de red es bastante simple, puesto que solo cuenta con dos capas de neuronas:

1. Capa de entrada: Formada por N neuronas, una por cada dato de variable de entrada. Su finalidad es la de recibir y transmitir la información a la capa de salida.

2. Capa de salida: Formada por M neuronas, procesa la información y forma el mapa de características que, normalmente, tiene una organización bidimensional. Las neuronas de salida tienen asociado un vector de pesos llamado vector de referencia, que constituye el vector prototipo o promedio de la categoría representada por la neurona de salida.

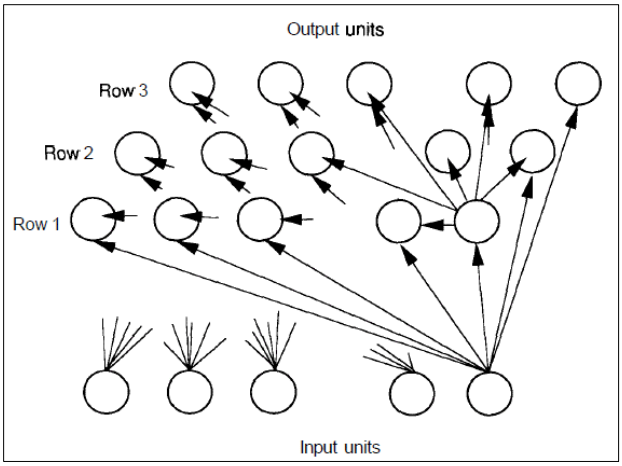

Figura 6. Representación de la arquitectura de una red de Kohonen (Fuente: Freeman y Skapura 1991)

Entre las neuronas de salida existen conexiones de excitación o inhibición con sus vecinas como se puede ver en la Figura 6, de manera que cada una de estas ejerce cierta influencia en aquellas más cercanas, aunque no estén conectadas. El grado con el que una neurona afecta a las vecinas dependerá de la aplicación de una función denominada función de vecindad (Kohonen 1989).

El problema que tiene esta característica de la red es que el mapa de neuronas debe mantener un orden, pues la alteración de la colocación de las neuronas puede afectar a la precisión del resultado.

El algoritmo general de la red de Kohonen se puede definir de manera sencilla en tres pasos principales:

1. Inicializar el mapa de neuronas *j* con vectores de pesos *W* aleatorios. Los valores estarán en el intervalo (0,1).
2. Tomar un vector de entrada, y por cada neurona del mapa:
    a) Calcular la distancia entre el vector de entrada *X* y los vectores de pesos *W*. Esta se calcula normalmente con la distancia euclidiana, definida por la ecuación:
    $$\left|\left|X - W_j\right|\right| = \sqrt{\sum(x_i - w_{ji})^2}$$
    b) Actualizar la neurona vencedora *G(X)*, aquella cuya distancia sea la de menor valor.
    $$G(X) = \min_j\{\left|\left|X - W_j\right|\right|\}$$
    c) Actualizar las neuronas en la vecindad de la neurona seleccionada. Para ello se describe la función de vecindad *h(t)* (Cottrell et al. 2018), donde *dist* es la distancia euclidiana entre ambas neuronas y $\sigma^2(t)$, que puede reducir la excitación por la vecindad a lo largo del tiempo.
    $$h_{j,G(X)}(t) = \exp(-\frac{dist^2{}_{j,G(X)}}{2\sigma^2(t)})$$
3. Incrementar la iteración y volver al paso 2 mientras queden iteraciones.

Red Counterpropagation

La red Counterpropagation (CPN, del inglés *Counterpropagation Network*), propuesta por R. Hecht-Nielsen, se trata de una red neuronal híbrida, es decir, que combina dos estructuras ya existentes: una SOM y una Outstar. Como se ha comprobado en apartados anteriores, ambas redes aplican distintos tipos de aprendizajes, lo que le permite aprender con mayor rapidez con respecto a otros tipos de redes con mismo o mayor número de capas (Freeman and Skapura 1991).

La arquitectura de la Counterpropagation cuenta, así representadas en la Figura 7, con tres capas:

1. Capa de entrada: Formada por una neurona por cada dato de variable de entrada. Realiza un preprocesado de los datos para normalizarlos antes de transmitir la información procedente del exterior.

2. Capa intermedia u oculta: Formada por tantas neuronas como clases se quiera clasificar. Siguiendo el modelo de Kohonen, cada neurona responde con un valor ganador ante un cierto grupo de vectores de entrada perteneciente a una región distinta del espacio, de manera que esta es la encargada de reconocer las características. La neurona con la mayor respuesta es la única que comunica un valor distinto de cero a la próxima capa.

3. Capa de salida: Las neuronas de la capa de salida se forman en estructuras Outstars. Existe una Outstar por cada Instar de la capa intermedia, más una neurona adicional oculta. Su objetivo es etiquetar a la clase que contiene la característica reconocida.

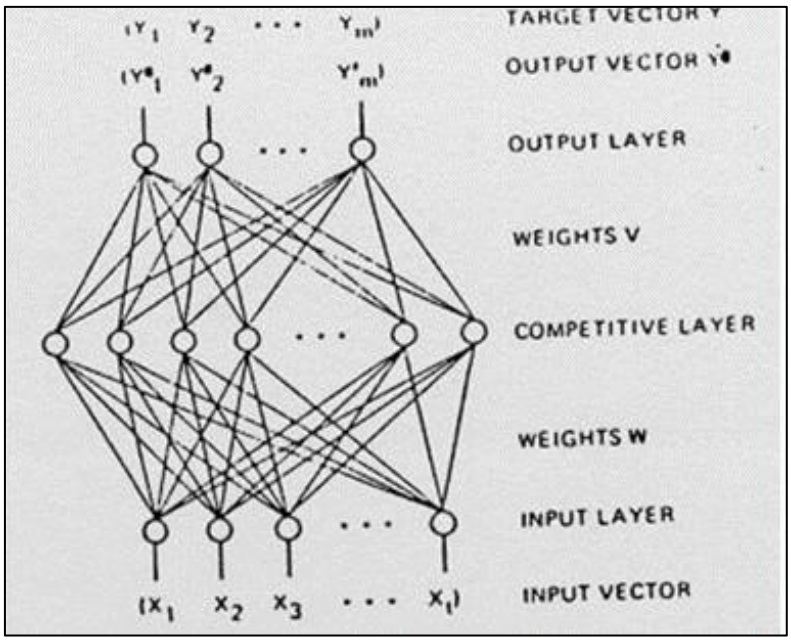

Figura 7. Estructura de una red Counterpropagation. Fuente:(Xabier Basogain Olabe 2008).

Dado que la CPN está formada por dos redes, su algoritmo de entrenamiento se limita a los tipos de aprendizajes de cada capa, lo que simplifica el funcionamiento de la red (R. Hecht-Nielsen 1987). En este caso, con la naturaleza de la red SOM y de la red Outstar dependiente de la estructura vecinal de los nodos, lo más importante es el ajuste de los valores de los pesos.

Es importante apuntar que, durante el entrenamiento de cada capa, los pesos del resto no se modifican. Así pues, el algoritmo de aprendizaje se puede resumir en los siguientes pasos (Xabier Basogain Olabe 2008):

1. Se normaliza el vector de datos de entrada *X* y su correspondiente vector de salida *Y*.
2. Se escoge un vector de entrada para inicializar los pesos *W* de las capas. Esta elección tiene como objetivo evitar el estancamiento de los pesos, situación que podría suceder en caso de hacer una inicialización aleatoria.
3. Se entrena la capa intermedia SOM, y se ajustan los valores de los pesos W. La ecuación que define el ajuste se define a continuación, donde *x* es el vector de entrada de la red, *w* son los pesos a la neurona ganadora y α es el coeficiente de aprendizaje para esta capa. Tras el ajuste, es necesario normalizar nuevamente el vector *w.*
$$\Delta W_i = \alpha(x_i * w_i)$$
4. Se entrena la capa de salida, y se ajustan los valores de los pesos *V*. Esta se realiza según la regla de Widrow-Hoff (Milin et al. 2020), cuyo objetivo es minimizar la diferencia del error cuadrático medio entre dos vectores de datos. Esta viene expresada por la siguiente ecuación, donde *y* es el vector de salida de la red, *y'* la salida esperada, β es el coeficiente de aprendizaje para esta capa y *Z* es la activación de las neuronas de la capa de Kohonen.
$$\Delta V_i = \beta Z_i(y_i * y'_i)$$

Ensembles neuronales

Un ensemble es un sistema que agrupa múltiples modelos o clasificadores de manera estratégica para resolver un problema particular por medio de computación inteligente. El objetivo principal de un ensemble es la mejora del rendimiento de un clasificador pobre (Robi Polikar 2009).

A diferencia de una red neuronal híbrida, que combina los clasificadores para que trabajen juntos como modelos interconectados, en un ensemble los clasificadores son independientes entre sí. Cada uno entrenará con sus datos y generará su salida. El ensemble se encargará luego de aunarlas en una respuesta final por medio de la aplicación de las reglas de combinación implementadas.

Los ensembles son especialmente útiles cuando se tiene una cantidad de datos enorme o con tipos de datos muy diversos, pues se puede dividir entre los distintos clasificadores para que se especialicen en menos características, o cuando se tiene una cantidad de datos muy pequeña, lo que permite aplicar una estrategia de tratado de los datos denominada *bootstrapping* para crear subconjuntos de datos con reemplazamiento. Como se puede intuir de este *modus operandi*, los clasificadores individuales que conformarán el ensemble se convertirán en clasificadores de un único espacio de los datos.

Para que un ensemble se forme de manera correcta y se obtenga una mejora con respecto a los clasificadores, es necesario que cada uno tenga una respuesta distinta que no solape con los demás y la regla correcta para seleccionar la mejor solución posible. Para eso, es muy importante el estudio de las estrategias de diversidad y de combinación.

Diversidad

La estrategia de diversidad es la más importante, puesto que el éxito del ensemble depende completamente de esta. Esto es así puesto que el ensemble corrige los errores realizados por los clasificadores, y si se diera que todos fallan igual entonces el ensemble no sería capaz de detectar el fallo para corregirlo.

Existen varias maneras de lograrlo. Una estrategia muy común es la de entrenar con distintos conjuntos de datos utilizando algoritmos con remuestreo de datos, creando así clasificadores pobres. También se puede alcanzar cambiando los parámetros de entrenamiento de los clasificadores, o utilizando clasificadores de tipos distintos, como árboles de decisión y máquina de vector de soporte (Báez et al. 2012). Se puede concluir que los clasificadores miembros de un ensemble más diversos son aquellos cuyas correlaciones son más bajas.

Combinación

La segunda estrategia es la de combinación de salidas. Esta asegura que el ensemble elegirá las decisiones correctas por encima de las incorrectas. Se pueden considerar dos maneras de combinación (Báez et al. 2012):

- Entrenables y no entrenables: Los métodos de combinación entrenables se caracterizan por necesitar un algoritmo de entrenamiento aparte que combinen los pesos entrenados por la red tras obtener su decisión, mientras que las estrategias no entrenables requieren parámetros obtenidos directamente del clasificador tras el entrenamiento.
- Etiquetas de clase y etiquetas continuas de salida: En este caso, el primer método hace uso de los valores que va obteniendo el clasificador y que representan el grado de estimación de cada clase. Entre los métodos más usuales de este tipo están el de votación por mayoría simple (SMV, del inglés *Simple Majority Voting*) y el de votación por mayoría ponderada (WMV, del inglés *Weighted Majority Voting).* Por otro lado, el último método de combinación requiere de reglas algebraicas o esquemas de decisiones para determinar el resultado final.

Algunos algoritmos de uso común en el ensamblado de clasificadores

Existen un buen número de algoritmos específicos para su utilización con los ensembles. En esta sección vamos a comentar tres de los algoritmos más utilizados, tanto por su sencillez como por su extensión de uso. Algunos de estos algoritmos ya cuentan con sus estrategias de diversidad y combinación propias.

Random Subsampling

El *Random Subsampling* es una técnica de selección de subconjuntos de datos a partir de un conjunto mayor para el entrenamiento de un clasificador (Hajare Akash 2021). Se caracteriza por realizar *k* iteraciones del conjunto entero, escogiendo cada vez un número de observaciones sin reemplazamiento como se puede observar en la Figura 8.

Cuando se ha entrenado el modelo con cada subconjunto, se habrán obtenido *k* clasificadores con sus espacios de error independiente. Es posible utilizar las respuestas obtenidas para combinarlas por medio de una decisión de votación, tanto con SMV como WMV.

Los pasos del algoritmo son los siguientes:

1. Para cada iteración *t*, seleccionar un número de observaciones *L* para el conjunto de testeo.
2. Se entrena el modelo y se estima el error con el conjunto de testeo.
3. Una vez acabado el entrenamiento, se halla el error medio predicho (*PE*).

$$PE = \frac{1}{T}\sum_{i=1}^{T} E_i$$

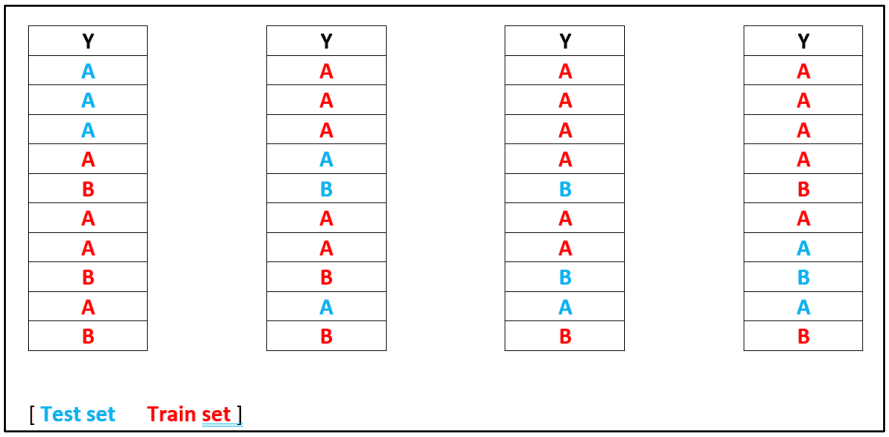

Figura 8. Proceso del algoritmo Random Subsamplig. Fuente: (Hajare Akash 2021)

Bagging

El *bagging* es una de las primeras técnicas aplicadas. Se trata de la más simple e intuitiva de todas, y es capaz de obtener un gran rendimiento. Para ello, hace uso del *bootstrapping*, una técnica utilizada para entrenar a los clasificadores con subconjuntos aleatorios con reemplazamiento a partir del conjunto original.

La idea de este algoritmo es sencilla. Se crean múltiples subconjuntos y se entrena el clasificador con cada uno de ellos, y se obtiene el promedio de las salidas de cada componente tal y como aparece representado en la Figura 9. Utilizando este método para combinar las salidas, aunque "no se obtiene un cambio de la respuesta esperada, se reduce su varianza" (Joseph Rocca 2019). La manera en la que se combinan estas salidas puede ser por SMV o WMV.

Los pasos principales del algoritmo se definen de la siguiente manera:

1. Seleccionar un porcentaje P para el método *bootstrap.*
2. Para cada iteración t, coger un subconjunto S_t aleatoriamente del conjunto total de datos, cumpliendo la proporción *P*.
3. Entrenar el clasificador con el subconjunto.
4. Una vez obtenidas todas las decisiones con todos los datos, combinarlos:

$$(SMV)\ H_T = \frac{1}{T}\sum_{t=1}^{T} w_t$$

$$(\text{WMV})\ H_T = \arg_k \max[card(t \mid w_t = k)]$$

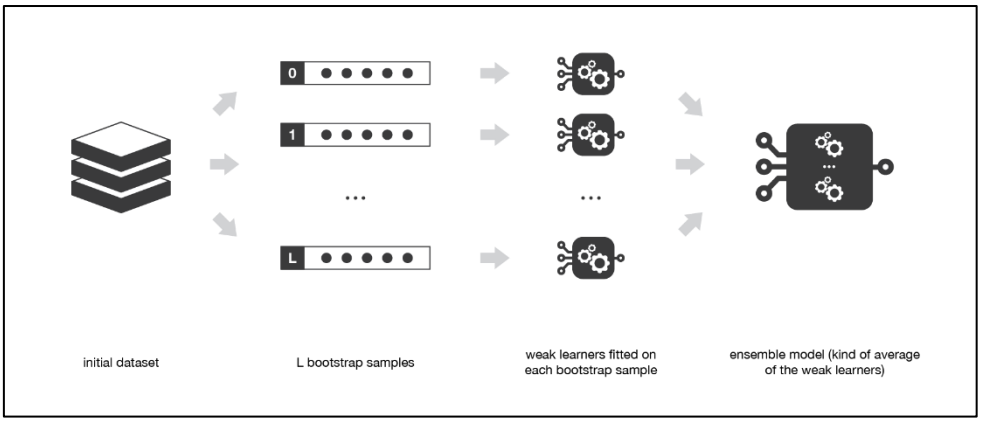

Figura 9. Proceso del algoritmo de Bagging. Fuente: (Joseph Rocca 2019)

Boosting

El *boosting* genera los subconjuntos también de manera muy sencilla, aunque es más inteligente que el algoritmo *bagging*. La idea principal es la de entrenar los clasificadores de manera sucesiva, haciendo que el entrenamiento del actual dependa de los modelos anteriores, como se puede apreciar en la Figura 10.

El enfoque de este algoritmo es reducir la parcialidad de los clasificadores débiles que lo componen, por lo que normalmente se aplica a modelos con una varianza baja. Las respuestas de los clasificadores se combinan normalmente por medio de una decisión de WMV.

Dado que es un algoritmo que depende del número de clasificadores que se tengan, los pasos que se darán a continuación serán un ejemplo específico para tres clasificadores (Robi Polikar 2009):

1. Seleccionar un subconjunto S_1 de datos.
2. Entrenar el clasificador C_1 con el subconjunto.
3. Crear un nuevo subconjunto S_2 haciendo que al menos la mitad haya sido clasificada correctamente por C_1, y la otra mitad esté mal clasificada.
4. Entrenar el clasificador C_2 con S_2.
5. Crear un subconjunto S_3 para C_3 compuesto por los datos en los que C_1 y C_2 no hayan dado la misma respuesta.
6. Entrenar el clasificador C_3 con S_3.

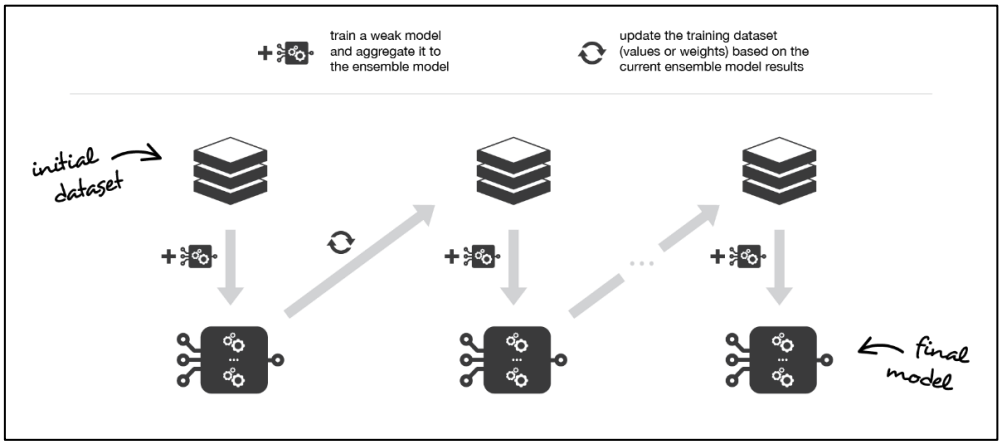

Figura 10. Proceso del algoritmo de Boosting. Fuente: (Joseph Rocca 2019)

AdaBoost

AdaBoost (del inglés *Adaptive boosting*), es un algoritmo que extiende Boosting para convertirlo en multiclase y aplicarlo a problemas de regresión. Dado que el resto de los algoritmos se encuentran limitados a problemas de clasificación binaria, AdaBoost presenta ventajas haciendo uso práctico de la misma regla de combinación con mejores resultados (Yoav Freund and Robert E. Schapire 1996).

Para cada clasificador, se obtiene un subconjunto de datos que se va actualizando en cada iteración del algoritmo. Como se puede ver en la Figura 11, la distribución de los datos comienza siendo uniforme, de manera que todos tienen la misma probabilidad de aparecer en el primer subconjunto, y se ajusta para el siguiente clasificador teniendo en cuenta el error cometido.

Una vez acabado el entrenamiento, el ensemble combina las respuestas por medio de una decisión de votación por mayoría ponderada.

Así pues, los pasos principales de este algoritmo (Yoav Freund and Robert E. Schapire 1996) son los siguientes:

1. A partir de un conjunto con *N* observaciones *(X,Y)*, crear la distribución *D* a partir de los ejemplos e inicializar el vector de pesos *w.* Para aplicar este método, los valores de *Y* deben mantenerse entre -1 y 1.
$$D_1(i) = 1/m$$
2. Para cada iteración *t*, construir un clasificador con la distribución p_i^t, obtenida de la normalización de los pesos, para obtener la hipótesis h_t.
$$p^t = \frac{w^t}{\sum_{i=1}^{N} w_i^t}$$
$$h_t : X \to [0,1]$$
3. Calcular el error asociado al clasificador de la hipótesis.
$$\varepsilon_t = \sum_{i=1}^{N} p_i^t |h_t(x_i) - y_i|$$
4. Obtener el parámetro para la actualización del vector de pesos.
$$\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}$$
5. Actualizar el vector de pesos.
$$w_i^{t+1} = w_i^t \beta_t^{1-|h_t(x_i)-y_i|}$$
6. Construir el clasificador final H_{final} como el promedio ponderado de los clasificadores.
$$\beta_t = \begin{cases} 1, & \sum_{t=1}^{T} \frac{\log 1}{\beta_t} \; h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \frac{\log 1}{\beta_t} \\ 0, & otro\ caso \end{cases}$$

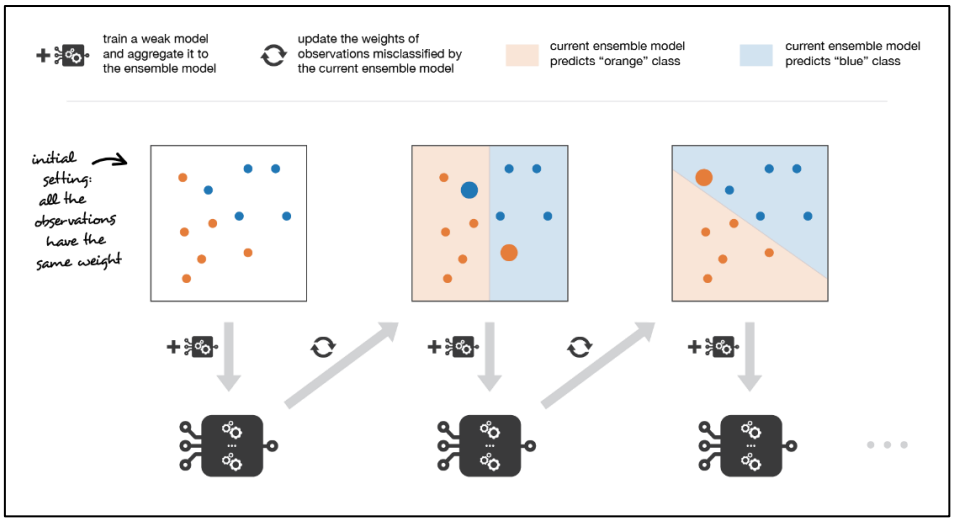

Figura 11. Proceso del algoritmo de AdaBoost. Fuente: (Joseph Rocca 2019)

Capítulo 2: Diseño y desarrollo de sistemas inteligentes de ayuda a la detección de la enfermedad de Alzheimer

En este capítulo se detallan los sistemas de detección diseñados con redes neuronales híbridas y ensembles neuronales. Primero se explicarán los conjuntos de datos seleccionados, las mediciones de calidad del clasificador y los métodos utilizados. Seguidamente, se especificarán las características la herramienta de MatLab que variarán durante las pruebas y se discutirá la implementación.

Conjunto de Datos

Los dos conjuntos de datos seleccionados para este proyecto, a los que hemos llamado conjunto A y conjunto B, proceden de Alzheimer's Disease Neuroimaging Initiative (ADNI), una iniciativa lanzada en 2004 para realizar un estudio longitudinal en varios centros. Los objetivos de ADNI (Michael W. Weiner 2017) son principalmente:

- La detección de la Enfermedad de Alzheimer en el estadio más temprano posible y la identificación de nuevas formas de análisis de la enfermedad por medio de biomarcadores.
- Fomentar el avance en la intervención, prevención y tratamiento de la EA aplicando nuevos métodos de diagnóstico.
- La administración del acceso a los datos de ADNI, proveyendo a los científicos de ellos sin embargos posteriores por su uso.

ADNI tiene cuatro conjuntos de datos que siguen la evolución de la enfermedad en determinados periodos de tiempo, añadiendo nuevos pacientes y pruebas de manera incremental con cada uno (Michael W. Weiner 2017). En la Figura 12 se encuentra una línea de tiempo de la vigencia de cada conjunto.

El primer conjunto es ADNI 1, con una duración de 6 años. Tiene un total de 800 sujetos: 400 con Deterioro Cognitivo Leve (DCL), 200 con Enfermedad de Alzheimer y 200 sanos de control (CN). Este contiene neuroimagen, perfiles genéticos, análisis de sangre y biomarcadores en el fluido cerebroespinal.

ADNI GO duró 2 años, y fue creado con el objetivo de examinar los biomarcadores en estadios tempranos de la demencia. Añade 200 sujetos con DCL a ADNI 1.

ADNI 2 añade 150 sujetos de control, 100 con DCL temprano, 150 con DCL tardío y 150 con EA. Además, se creó un nuevo grupo de 107 sujetos con Preocupación Significativa de Memoria (SMC, del inglés *Significant Memory Concern*) y una nueva prueba de neuroimagen, con el fin de desarrollar biomarcadores que permitan la predicción del declive cognitivo.

Finalmente, el conjunto de datos vigente hasta la fecha es ADNI 3. Este busca la relación entre las distintas pruebas clínicas, cognitivas, genéticas, de neuroimagen y biomarcadores a lo largo de la EA. Añade escaneos cerebrales para detectar la proteína tau, un indicador clave.

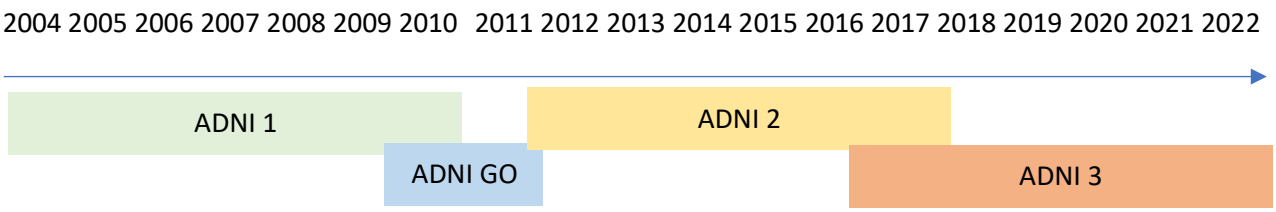

Figura 12. Línea de tiempo de los conjuntos de datos de ADNI.

De estos distintos *datasets*, se han extraído dos subconjuntos de ADNI 2, conjunto A y conjunto B, con ayuda de expertos médicos en contacto con el Grupo de Investigación de Computación Neuronal y Adaptativa y Neurociencia Computacional (COMCIENCIA). Cada conjunto cuenta con dos valoraciones de factores de riesgo (edad y nivel educacional) y tres pruebas cognitivas breves.

El primer test cognitivo, compartido por ambos conjuntos, es el Cuestionario de Actividades Funcionales (FAQ, del inglés *Functional Activities Questionnaire*). Este mide las actividades rutinarias del paciente, monitorizando los cambios a través del tiempo, por lo que es útil para determinar si padece DCL o EA leve (Edmond Teng 1, Brian W Becker, Ellen Woo, David S Knopman, Jeffrey L Cummings, Po H Lu 2010).

La segunda prueba utilizada en el conjunto A es el Mini Mental de Folstein (MMSE, del inglés *Mini-mental state*). Se trata de una prueba estandarizada para determinar si el paciente padece de una demencia. Además, permite seguir su evolución dependiendo de la puntuación obtenida (M F Folstein, S E Folstein, P R McHugh 1975).

La última prueba del conjunto A es la Escala de Depresión Geriátrica (GDS, del inglés *Geriatric Depression Scale*). Esta prueba es capaz de detectar la depresión en los mayores. Su utilidad radica en la capacidad que tiene para distinguirla en personas que sufren demencia en estado leve hasta moderado (Sheikh, Javaid I. Yesavage, Jerome A. 1986).

La Evaluación Cognitiva de Montreal (MoCA, del inglés *Montreal Cognitive Assessment*) es la segunda prueba del conjunto B. Evalúa las disfunciones cognitivas leves dividiendo ocho habilidades en preguntas rápidas (Ziad S Nasreddine 1, Natalie A Phillips, Valérie Bédirian, Simon Charbonneau, Victor Whitehead, Isabelle Collin, Jeffrey L Cummings, Howard Chertkow 2005).

Finalmente, la última prueba cognitiva es el Inventario Neuropsiquiátrico (NPI, del inglés *Neuropsychiatric Inventory*). A diferencia de las otras, esta permite realizar un seguimiento de la eficacia de los tratamientos clínicos sobre los pacientes con demencia, determinando la gravedad según el número de trastornos sufridos con frecuencia (J L Cummings 1, M Mega, K Gray, S Rosenberg-Thompson, D A Carusi, J Gornbein 1994).

A lo largo del estudio, se renombrará la edad a su término inglés *AGE*, y los años de educación tendrán un nombre distinto en cada conjunto. El conjunto A se referirá a este factor con el nombre de PTEDUCAT, sacado directamente de la base de datos ADNI; el conjunto B lo abreviará con su término inglés *EDUCATION* simplemente. Por otro lado, la medida de NPI utilizada será la total del paciente, denominada NPITOTAL.

El conjunto A cuenta con un total de 331 sujetos: 128 con DCL y 203 de control. Por otro lado, el conjunto B cuenta con 462 sujetos: 296 con Deterioro Cognitivo Leve y 166 de control como se ve en la Tabla 5. Ninguno de los conjuntos de datos se encuentra balanceado en las

clases como se puede comprobar en la Tabla 4, puesto que hay una gran diferencia entre los sujetos de la clase DCL y los de CN.

Para la red se preparan dos subconjuntos de datos, uno para entrenamiento y otro para validación, de cada conjunto con una proporción de 76%-23% de los datos. De esta forma, se garantiza un buen entrenamiento con suficientes pacientes, pero sin descuidar el tener un conjunto mínimo para la validación.

Tabla 4. Estadística descriptiva del conjunto A.

Atributos	Control	DCL	Total
Sujetos	203	128	331
AGE			
Media ± Std[23]	74,1346 ± 6,2604	74,9177 ± 7,2344	74,4374 ± 6,6542
Intervalo	[56,2521 - 89,0658]	[56,2548 - 88,0192]	[56,2521 - 89,0658]
MMSE			
Media ± Std	29,0640 ± 1,2027	27,1953 ± 1,7346	28,3414 ± 1,6953
Intervalo	[24,0000 - 30,0000]	[24,0000 - 30,0000]	[24,0000 - 30,0000]
FAQ			
Media ± Std	0,1724 ± 0,6252	3,6406 ± 4,4239	1,5136 ± 3,2607
Intervalo	[0,0000 - 5,0000]	[0,0000 - 20,0000]	[0,0000 - 20,0000]
GDS			
Media ± Std	0,7931 ± 1,1717	1,5703 ± 1,4509	1,0937 ± 1,3395
Intervalo	[0,0000 - 6,0000]	[0,0000 - 5,0000]	[0,0000 - 6,0000]
PTEDUCAT			
Media ± Std	16,5222 ± 2,6087	15,4844 ± 3,1920	16,1208 ± 2,8884
Intervalo	[10,0000 - 20,0000]	[4,0000 - 20,0000]	[4,0000 - 20,0000]

[23] "Std" corresponde a la Desviación Típica y define el error producido en la media. Esta estadística se interpreta como el grado de "dispersión media de una variable" (José Francisco López 2017).

Tabla 5. Estadística descriptiva del conjunto B.

Atributos	Control	DCL	Total
Sujetos	166	296	462
AGE			
Media ± Std	73,5584 ± 6,0790	71,4645 ± 7,1588	72,2169 ± 6,8580
Intervalo	[59,9000 - 89,0000]	[55,0000 - 88,6000]	[55,0000 - 89,0000]
EDUCATION			
Media ± Std	16,4458 ± 2,5238	16,2264 ± 2,6970	16,3052 ± 2,6355
Intervalo	[12,0000 - 20,0000]	[9,0000 - 20,0000]	[9,0000 - 20,0000]
MOCA			
Media ± Std	25,7349 ± 2,3566	23,1115 ± 3,1812	24,0541 ± 3,1704
Intervalo	[19,0000 - 30,0000]	[14,0000 - 30,0000]	[14,0000 - 30,0000]
FAQ			
Media ± Std	0,1687 ± 0,6193	2,9966 ± 3,9320	1,9805 ± 3,4462
Intervalo	[0,0000 - 5,0000]	[0,0000 - 20,0000]	[0,0000 - 20,0000]
NPITOTAL			
Media ± Std	0,8253 ± 2,2183	4,0101 ± 6,2189	2,8658 ± 5,3712
Intervalo	[0,0000 - 16,0000]	[0,0000 - 33,0000]	[0,0000 - 33,0000]
HIPPOCAMPUS			
Media ± Std	7497,3735 ± 876,9188	6996,7365 ± 1106,5906	7176,6190 ± 1056,7205
Intervalo	[5160 - 9700]	[3914 - 9902]	[3914 - 9902]

Selección de atributos

Cuando se entrena un clasificador, la información recibida debe estructurarse para el tipo de sistema que vaya a procesarla. Además, es necesario elegir la mejor representación de datos para obtener la mejor respuesta del sistema (Jason Brownlee 2020a). Esto es especialmente importante cuando se tiene un gran número de variables o atributos que forman parte del problema, con mayor y menor relevancia para la resolución.

Si se utilizaran todos los datos directamente y hubiera muchos más atributos con poca relevancia, tendría como resultado un modelo cuyas predicciones no son de confianza. Para evitar esto, se hace una selección de los atributos más significativos.

Existen tres técnicas aplicables para la selección de atributos más significativos (Vikashraj Luhaniwal 2020):

- Métodos de filtrado: Son genéricos, puesto que no implican un algoritmo de aprendizaje específico.
- Método *Wrapper*: Evalúa el resultado de los atributos en un algoritmo de aprendizaje concreto. Este es el método implementado en el sistema de detección con redes neuronales híbridas.
- Método *Embedded*: Escoge los mejores atributos tras ver cómo afectan durante el proceso de aprendizaje.

Método *Wrapper*

Para aplicar el *wrapper*, es necesario tener un algoritmo de aprendizaje seleccionado de antemano. Este consiste en un bucle que va cogiendo todas las combinaciones posibles de atributos y entrenando el algoritmo en cada iteración y obteniendo la evaluación (Figura 13).

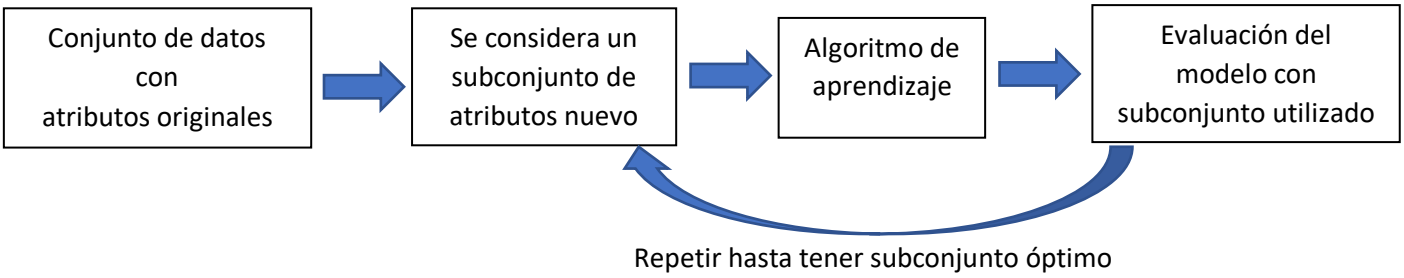

Figura 13. Descripción de los pasos para aplicar el método Wrapper (Fuente: Vikashraj Luhaniwal 2020).

Esto resulta en que, por cada configuración, el número de resultados dependerá de cuántos atributos se manejen en los datos del problema, haciendo que aumente el tiempo de procesamiento considerablemente a mayor número de atributos.

Esto se convierte en una combinatoria sencilla, calculada de la siguiente forma:

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

Sistemas Inteligentes para ayuda a la detección de la EA

En este TFG se han desarrollado dos Sistemas Inteligentes para ayuda a la detección de la EA, uno basado en las redes neuronales híbridas Counterpropagation y otro basado en ensembles neuronales. En este apartado se describirá el modelo de ambos sistemas.

Counterpropagation

La Figura 14 describe el sistema basado en Counterpropagation. Este recibe los datos y los preprocesa, adecuándolos al normalizado necesario para poder realizar el entrenamiento con este tipo de red.

Se han entrenado varias configuraciones diferentes de sistemas, que difieren en los valores de los parámetros descritos en la sección de Configuración del modelo. Para cada configuración individual, se ha entrenado mediante el método de *wrapper* para la selección de los atributos óptimos.

Una vez obtenidas todos los resultados de las combinaciones de atributos de una configuración, son analizados.

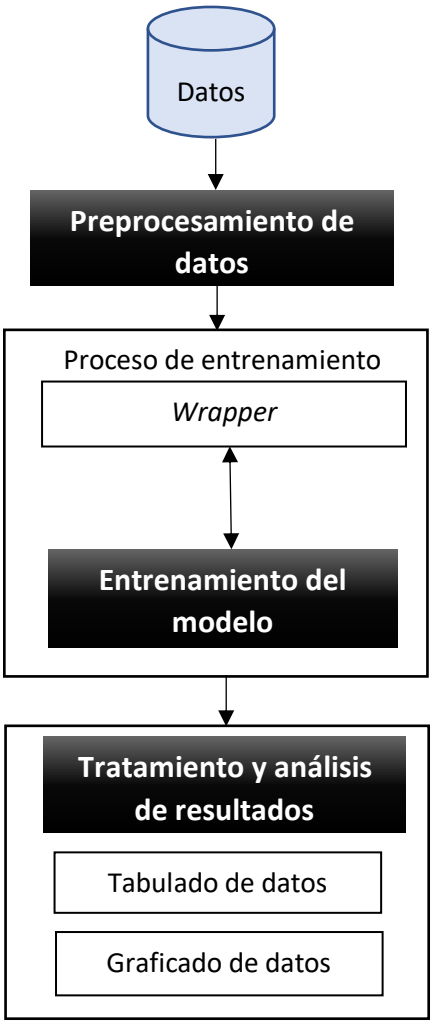

Figura 14. Diagrama del sistema basado en Counterpropagation.

Ensembles neuronales

La Figura 15 describe el sistema basado en ensembles neuronales construidos con Counterpropagation. Este recibe los datos y los preprocesa, adecuándolos al normalizado necesario para poder realizar el entrenamiento con este tipo de red.

Se define un número de clasificadores miembros del ensemble a partir del número de configuraciones aportadas al sistema. Estos clasificadores son elegidos en base al mayor grado de diversidad entre ellos, es decir, que tengan una menor correlación de los resultados.

Se entrenan y luego se combinan las decisiones de todos estos clasificadores por medio de las técnicas de combinación SMV o WMV.

Una vez obtenidas todos los resultados de las combinaciones de atributos de una configuración, son analizados.

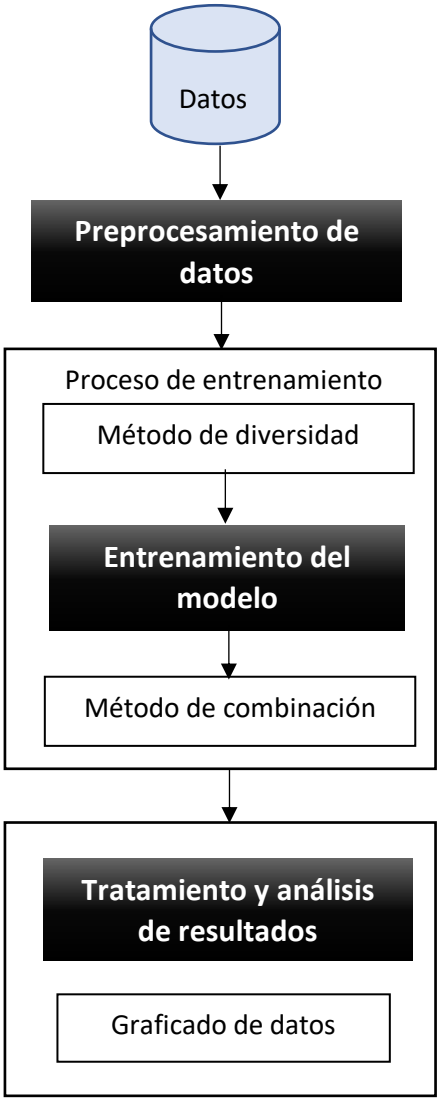

Figura 15. Diagrama del sistema basado en ensembles neuronales.

Medición de calidad de los clasificadores

Dado que el objetivo final de un clasificador es elaborar un modelo que permita aprender y predecir a partir de unos datos, los clasificadores cuentan con una alta dependencia sobre estos. Si aplicamos los modelos en una cuestión en la que las decisiones son de suma importancia, como se da en el campo médico al determinar si un paciente padece o no una enfermedad, el buen funcionamiento se vuelve crítico.

Para seleccionar el modelo más adecuado, se utilizan distintos tipos de métricas que dan la información suficiente para evaluar un clasificador como bueno o malo para su tarea.

Matriz de confusión

"La matriz de confusión es una tabla que resume el nivel de éxito de las predicciones de un modelo de clasificación" (Google Developers 2020). Esto quiere decir que su objetivo es mostrar el número de muestras etiquetadas por el modelo para cada clase, como se aprecia en el ejemplo de la Tabla 6.

Para un clasificador binario, que decide si un dato es de una clase o no lo es, de esta matriz se obtienen:

- Positivos verdaderos (TP, del inglés *True Positives*): Las predicciones determinan que el dato pertenece a la clase, y coincide correctamente.
- Positivos falsos (FP, del inglés *False Positives*): Las predicciones determinan que el dato pertenece a la clase, pero no es correcto.
- Negativos verdaderos (TN, del inglés *True Negatives*): Las predicciones determinan que el dato no pertenece a la clase, y efectivamente no lo es.
- Negativos falsos (FN, del inglés *False Negatives*): Las predicciones determinan que el dato no pertenece a la clase, pero sí lo es.

Tabla 6. Matriz de confusión de ejemplo para un clasificador binario.

Clases	Positivos	Negativos
DCL	32	6
CN	22	51

Error

El porcentaje de error es uno de los indicadores principales de un clasificador neuronal, definiendo la cantidad de datos que no ha sido capaz de clasificar tras el entrenamiento. Como explica Jason Brownlee en una entrada del blog Machine Learning Mastery (Jason Brownlee 2020b), es el indicador perfecto para evitar que un clasificador sufra de sobreajuste (*overfitting),* causando que se especialice demasiado en los datos y sea incapaz de predecir el resultado con otros conjuntos.

Puede tratarse de una función elaborada o de un simple cálculo matemático a partir de los valores obtenidos de la matriz de confusión.

Accuracy

Otra medida muy importante es el *accuracy* de la red. Con ella se mide la exactitud que tiene un clasificador al predecir nuevas muestras de datos, es decir, da el porcentaje de decisiones que el modelo ha realizado correctamente (Google Developers 2020).

$$\text{Accuracy} = \frac{Número\ total\ de\ predicciones\ correctas}{Número\ total\ de\ predicciones} = \frac{TP+TN}{TP+TN+FP+FN}$$

Especificidad y Sensibilidad

La especificidad y la sensibilidad son dos medidas que están altamente relacionadas, puesto que representan la tasa de aciertos realizados por el clasificador, tanto al decidir que el dato pertenece a la clase como no (Google Developers 2020).

Ambas se utilizan para el cálculo de la curva ROC.

$$\text{Sensibilidad} = \frac{TP}{FN+TP} \qquad \text{Especificidad} = \frac{TN}{FP+TN}$$

Índice de utilidad clínica (CUI)

El índice de utilidad clínica (CUI, del inglés *Clinical Utility Index*) determina hasta qué punto las decisiones de un clasificador son apropiadas para un diagnóstico real. Este se calcula para clases positivas y para clases negativas como un producto de la especificidad con los valores positivos predichos y el producto de la sensibilidad con los valores negativos predichos (Mitchell, 2008), obteniendo la valoración para ambos tipos de decisiones.

Los valores del CUI oscilan entre 0 y 1, siendo el primero un índice que determina una utilidad nula para el diagnóstico, mientras que el segundo es completamente fiable. Para tener una mayor rigurosidad a la hora de interpretar los resultados de esta medida, se clasifica en cinco rangos (Mitchell 2008), que son los siguientes:

- Pobre: El valor está entre 0 y 0,2.
- Aceptable: El valor está entre 0,2 y 0,4.
- Moderado: El valor está entre 0,4 y 0,6.
- Bueno: El valor está entre 0,6 y 0,8.
- Muy bueno: El valor está por encima del 0,8.

$$\text{CUI-} = \text{Sensibilidad} * \frac{TP}{TP+FP} \qquad \text{CUI+} = \text{Especificidad} * \frac{TN}{TN+FN}$$

Curva ROC y AUC

La curva Característica Operativa del Receptor (ROC, del inglés *Receiver Operating Characteristic*) es una representación de la sensibilidad frente a la especificidad. Su objetivo es el de la "visualización, organización y selección de clasificadores basados en su rendimiento" (Fawcett 2006). Dados los parámetros utilizados para construirla, los valores debajo de la curva representan los falsos positivos (FPR, del inglés *False Positive Rate*) y el área sobre la curva representa a los verdaderos positivos (TPR, del inglés *True Positive Rate*).

El valor de TPR y FPR se calcula de la siguiente manera:

TPR = Sensibilidad FPR = 1 - Especificidad

Esta curva se obtiene con la variación de un umbral o *threshold* en la salida de la red, que modifica las decisiones finales del clasificador y, por tanto, hace que varíen la sensibilidad y la especificidad. La curva se mantiene entre los valores 0 y 1, de manera que el umbral debe alterarse entre estos dos valores para conseguir tantos puntos como se desee.

La curva ideal es aquella cuyo FPR es 0 y TPR es 1, como se aprecia en la Figura 16. También se suele representar la curva ROC de un clasificador aleatorio, que representa la diagonal, dado que podrá acertar al azar la mitad de las veces (Fawcett 2006). Un clasificador que esté por debajo de la diagonal se considera malo.

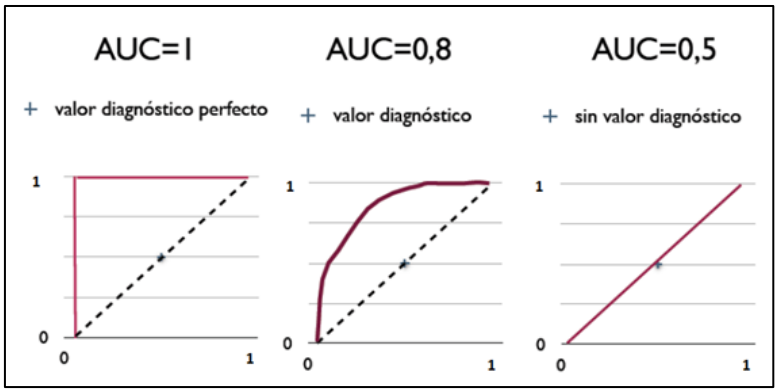

Figura 16. Representación de una curva ROC perfecta (1), una curva ROC buena (2) y una curva ROC aleatoria (3). Fuente: Wikipedia.

En el caso de un ensemble, la curva ROC se puede hallar de distintas maneras. Siguiendo la explicación de Ludmila Kuncheva, un ensemble está compuesto por varios clasificadores, lo cual hará que la respuesta del clasificador sea condicionada por los clasificadores individuales (Vilariño, Kuncheva, and Radeva 2006). Esta aproximación se puede realizar con:

- Combinado máximo: Dado que el ensemble tendrá en cuenta las salidas de todos sus clasificadores individuales y corregirá sus errores, la curva ROC será la combinación de los mejores valores de cada clasificador, como se puede observar en la Figura 17.

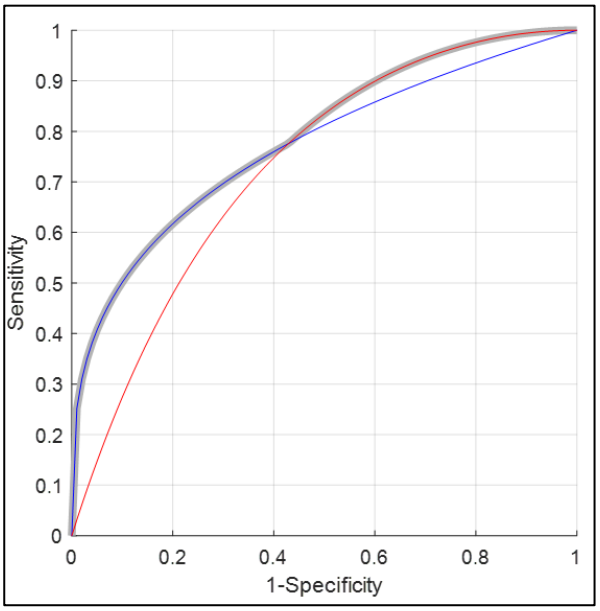

Figura 17. Ejemplo de la curva ROC resultante de la unión de los clasificadores rojo y azul (Vilariño, Kuncheva, and Radeva 2006).

- Combinado suavizado: La curva ROC del ensemble tendrá en cuenta todos los valores de los clasificadores individuales, permitiendo que se realicen decisiones no tan buenas para dar una curva más realista.

El Área Bajo la Curva (AUC, del inglés *Area Under the ROC Curve*) es la superficie que se encuentra por debajo de la curva ROC. Esta representa "la probabilidad de que el modelo clasifique un ejemplo aleatorio positivo más alto que un ejemplo negativo aleatorio" (Google Developers 2020).

El cálculo más sencillo para obtener el AUC es por medio de la regla del trapecio. Esta consiste en dividir el área bajo la curva en zonas con forma de trapezoide, de manera que la integración se realiza en áreas más manejables aplicando la fórmula del área de esta figura geométrica. En la Figura 18 se puede apreciar cómo se ha dividido el área en ocho zonas trapezoidales uniformes, con el objetivo de hallar área bajo una curva senoidal.

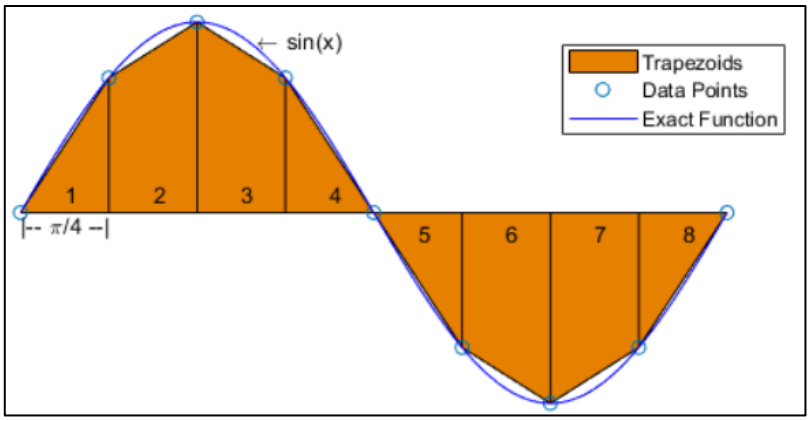

Figura 18. Representación del procedimiento de la regla del trapecio. Fuente: (MatLab 2006)

Este cálculo se realiza en el proyecto por medio de dos funciones:

- Trapz: Función propia de MatLab (MatLab 2006), para el ensemble neuronal. Recibe como parámetros los puntos en X y en Y de la curva ROC.
- Roc_curves: Función cedida por el profesor Patricio García Báez, que implementa el cálculo del área del trapecio bajo la curva aplicado a los pesos de los clasificadores. Esta elabora los puntos de la curva por medio de un umbral o *threshold* que varía en 0,1.

Estructuras de Kohonen and CPANN Toolbox

La herramienta *Kohonen and CPANN Toolbox* ofrece un conjunto de módulos para entrenar redes de tipo SOM, Counterpropagation y redes XY-fused. De todos estos, se ha utilizado de manera directa únicamente el módulo de Counterpropagation, que a su vez hace uso del SOM.

Las principales funciones usadas han sido:

- model_cpann: Construye un modelo de red Counterpropagation a partir de los datos de entrenamiento, el vector de etiquetas de clases y la configuración de la red.
- pred_cpann: Presenta un conjunto de validación al modelo previamente entrenado. Necesita el nuevo conjunto de datos y el modelo base.
- cpann_class_param: Obtiene los parámetros del clasificador para medir la calidad del sistema.

Adicionalmente, para hacer uso de la función roc_curves del profesor Patricio G. se ha accedido a determinadas estructuras de los modelos:

- model.net.W_out: Matriz de pesos de salida para cada clase clasificada del modelo entrenado.
- pred.top_map: Matriz de posiciones en el mapa de Kohonen de los datos predichos por la red.

Configuración del modelo

La arquitectura neuronal utilizada para el desarrollo del sistema de detección ha sido una red Counterpropagation. Para la configuración de la red, se ha optado por realizar un estudio de pruebas jerárquicas, comenzando con unos parámetros de base y modificando según los mejores resultados.

En este caso, se ha decidido poner el enfoque cuatro de los parámetros de la red:

- Tamaño (*nsize*): Es el número de neuronas, por cada lado, que tendrá la capa del SOM. Dado que esta arquitectura se forma con una topología cuadrada, la cantidad final de neuronas será *nsize x nsize*. Ejemplo: Si el tamaño es 6, la cantidad de neuronas será de 36 (6x6). Debido a esto, se ha intentado mantener este parámetro por debajo del supuesto de tener una neurona por dato de entrada. Así, se ha variado este parámetro en los siguientes rangos: 4, 8, 10, 12 y 14 para el conjunto A y 4, 8, 10 ,12 ,14 y 16 para el conjunto B.

- Épocas (*epochs*): Se define una época como el recorrido completo de todos los patrones de entrada en la red, y puede estar formada por múltiples iteraciones. Dado que el coste temporal aumenta paralelamente con este parámetro, se han limitado las pruebas al siguiente rango: 10, 100, 200, 500 y 1000.

- Entrenamiento (*training*): Algoritmo para el entrenamiento de los pesos de la red. Puede realizarse de forma secuencial ('sequential'), de manera que se actualiza el conjunto de pesos tras cada iteración, o por lotes ('batch'), que actualiza los pesos tras cada época. Por lotes se presenta como un método más rápido, por lo que el objetivo es estudiar qué variación se produce con este cambio.

- Límites del mapa (*bound*): La forma del borde del SOM puede ser '*normal*' o '*toroidal*'. Con un límite toroidal, todos los bordes del mapa están conectados entre sí, mientras que con el normal los extremos permanecerán desconectados. En la Figura 19 se observa cómo se puede transformar un mapa normal en uno toroidal al plegar los bordes. Esto afecta al entrenamiento, dado que los vecinos de una neurona activada cambian.

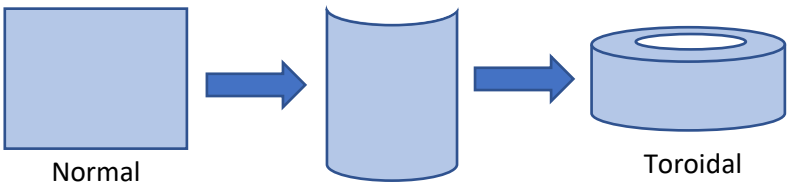

Figura 19. Representación de cómo un mapa 2D (normal) se convierte en una forma toroidal al plegarlo.

Por otro lado, la red necesita más parámetros que se han mantenido sin variación durante las pruebas. Estos han sido:

- Tipo de red (*som_settings*): Debido a que esta Toolbox permite utilizar la misma implementación para los mapas de Kohonen y las CPANNs, con este parámetro se especifica qué red se va a utilizar, siendo las posibilidades *'kohonen'* y *'cpann'*. Siguiendo el estudio de este proyecto, se utilizará la segunda opción.
- Topología (*topol*): Define la organización de las neuronas, de manera que pueden agruparse con topología cuadrada (*'square'*) o topología hexagonal (*'hexagonal'*). Esto

afecta directamente a la cantidad de vecinos, puesto que con una topología cuadrada las neuronas tienen 4 vecinos, mientras que con la topología hexagonal aumenta a 6 vecinos, como se puede apreciar en la Figura 20. Se ha elegido la topología hexagonal.

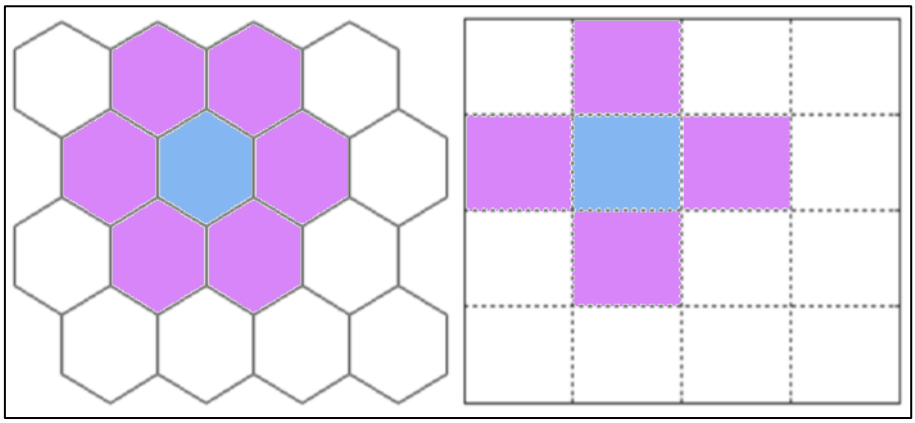

Figura 20. Representación de la topología hexagonal (izquierda) y de la cuadrada (derecha) de la red, con una neurona resaltada en azul y sus vecinos en violeta. (Basado en: Ballabio, Consonni, y Todeschini 2009)

- Inicialización (*init*): El tipo de inicialización para los pesos de la capa de Kohonen. Puede ser aleatoria (‘*random’*) o según los vectores característicos (‘*eigen’*) de las dos componentes principales más grandes. Este parámetro se ha fijado en ‘*eigen’*, puesto que la inicialización se realizará siempre con los mismos valores, asegurando que será el mismo siempre.
- Velocidad de aprendizaje inicial y final (*a_max* y *a_min*): Determinan el grado de corrección que se producirá en cada iteración del algoritmo. En este caso, se ha permitido todo el rango dejando el máximo a 1, y el mínimo a 0,1.
- Escalado (*scaling*): Preprocesado automático de los datos para que se puedan comparar correctamente con los pesos de la red. Por defecto, no se aplica (*‘none’*), pero ofrece centrar los datos (‘*cent’*), escalado (*‘scal’*) y ambos (*‘auto’*). Se ha mantenido el parámetro en *‘auto’*.
- Rango absoluto (*absolute_range*): Permite aplicar distintos tipos de escalados. Puede tener los valores 0, para que el escalado se aplique de manera independiente a cada columna del conjunto de datos, o 1, y se aplica en los valores absolutos máximos y mínimos de los datos. Dado que el valor a 1 es más apropiado en caso de tener datos que deban mantener la forma de relación (por ejemplo, en perfiles o espectros), se ha mantenido este parámetro a 0.
- Barra de progreso (*show_bar*): Si su valor es 0, mostrará el progreso de cada época entrenada. Si el valor es 1, será una barra de espera para el entrenamiento completo. Dado que no es necesario tener información de cada época y para obtener una mayor agilidad en la ejecución, el valor utilizado ha sido el 1.
- Método de asignación (*ass_meth*): El método utilizado para seleccionar a qué clase pertenece cada neurona. Tiene 4 valores posibles:
    - 1: Valor por defecto, las neuronas se asignan a la clase con el mayor peso de salida. Se ha dejado en este valor.
    - 2: La neurona se asigna si la diferencia entre el peso mayor y la siguiente mayor es mayor que un umbral.

- 3: Se asigna la clase a la neurona solo si el peso de salida mayor es más grande que un umbral.
- 4: La clase se asigna según una función de suavizado.

Con estos parámetros, y siguiendo el orden para aplicarlos al modelo, una configuración de ejemplo sería como la de la Tabla 7.

Tabla 7. Ejemplo de configuración para una red CPANN con la Toolbox de MatLab.

Som_settings	Nsize	Epochs	Topol	Bound	Training	Init
	6	300	Hexagonal	Toroidal	Batch	Eigen
Cpann	**A_max**	**A_min**	**Scaling**	**Absolute_range**	**Show_bar**	**Ass_meth**
	1	0.1	Auto	0	1	1

Implementación del sistema de detección

El sistema de detección se ha realizado con la ayuda de la implementación de la herramienta *Kohonen and CPANN Toolbox*. Este consta de dos partes diferenciadas en el uso de métodos computacionales: redes neuronales híbridas y ensembles neuronales. Para ello, se ha dividido el programa en los tres pasos principales que conforman el esqueleto de un algoritmo general en inteligencia artificial.

Requisitos

De cara a una buena implementación, se han definido una serie de requisitos mínimos funcionales que debe cumplir el sistema para presentar un buen objeto de estudio, como se puede observar claramente en la Figura 14 y la Figura 15.

- Preprocesamiento de los datos.
    1. Aunque se cuenta con los datos convenientemente tratados antes de que MatLab los lea, puesto que ya se encuentran tabulados por atributos, es necesario adaptar los conjuntos de datos para obtener los subconjuntos de entrenamiento y validación, así como adaptar estos a la entrada que requiere la red.
    2. Obtener de un fichero de texto todas las configuraciones de red que se desea probar y prepararlas con el formato necesario.

- Entrenamiento del modelo.
    1. Para el caso de la red neuronal híbrida CPN es necesario el método *wrapper,* que permitirá hacer la prueba de cada combinación posible de la configuración utilizada con el fin de encontrar la mejor posible.
    2. En el ensemble neuronal se necesitará un método para la generación de diversidad del modelo, que seleccionará los subconjuntos de datos con los que entrenarán los clasificadores individuales. Posteriormente, aún como parte del entrenamiento, se combinarán las respuestas con dos posibles métodos: SMV o WMV.
- Tratamiento y análisis de los resultados.

1. Tabulado de los resultados por configuraciones, permitiendo una comparación orgánica de todas las estadísticas de calidad.
2. Visualización gráfica de las curvas ROC que representan el rendimiento del sistema para cada combinación. En el caso del ensemble neuronal, además, se aplicarán todas las curvas de los clasificadores (ensemble y sus miembros).
3. Guardado de los resultados en archivos de MatLab para acceder a ellos desde el entorno siempre que sea necesario procesarlos después de la ejecución.

Ficheros principales

Los principales ficheros que componen los programas del sistema de detección se describen en este apartado:

- *sistemaCPN.m*: Es el *script* principal para el sistema de detección de la red neuronal híbrida. Para ejecutarlo es necesario configurar los archivos de conjuntos de datos que se utilizarán y crear un fichero denominado "configuracion.txt" con las configuraciones de red que se quiera estudiar en cada línea, separando cada parámetro con espacios. Inicialmente, parte de un directorio base desde el cual se obtendrá este fichero de texto y en el cual se irán volcando los resultados a medida que se vayan obteniendo.
- *neuralEnsemble.m*: El *script* principal para el sistema de detección con ensembles neuronales. Al igual que el anterior, requiere de los archivos de conjuntos de datos y el fichero "configuracion.txt". Adicionalmente, se debe proporcionar un archivo de atributos en el que se indiquen los atributos que cada clasificador individual utilizará.
- *trainingNet.m*: Este *script* engloba la versión más sencilla del entrenamiento del modelo. Está compuesto por una única función que entrena y valida la CPN, así como obtiene las estadísticas de calidad del modelo.
- *mostradoTest.m*: Este fichero que contiene solo una función, tiene el objetivo de graficar una curva ROC a partir de los valores FPR y TPR obtenidos del modelo. Con este fichero, se crean todas las curvas ROC de los sistemas de detección y, si se tiene los datos que necesita, se puede utilizar fuera de la ejecución natural del programa.

Funciones del núcleo del sistema de detección

En este apartado se mostrarán las funciones principales de los programas. A continuación, se explicará el funcionamiento breve de cada una de ellas, junto con un extracto de las partes más importantes.

- *wrapper(M,Mv,nombresAtributos, file, combination, settings, currentConfiguration, ruta,fileName)*

Esta función implementa el método wrapper para la selección de atributos con los que se entrenará la red CPN. Recibe como parámetros necesarios para el entrenamiento:

- *M*: Los datos de entrenamiento.
- *Mv*: Las etiquetas que determinan a qué clase pertenece cada fila de datos.
- *nombresAtributos*: La lista de atributos que forman el conjunto de datos completo, que se utilizará para el bucle principal de esta función.
- *settings* **:** Conjunto de parámetros necesarios para entrenar la red en la configuración.

El resto de los parámetros que recibe esta función son utilizados para el proceso de guardado, y serán explicados más adelante. Estos son: file, combination, currentConfiguration, ruta y fileName.

Es necesario advertir que esta función tiene una limitación importante en cuanto a la generación de las combinaciones. Según la documentación de MatLab, esta función produce una enumeración de todas las combinaciones posibles de los elementos del vector tomados a la vez. Esto implica que es una función costosa y, en un caso práctico, no se recomienda utilizar para más de 15 elementos. Esto implica que el *wrapper* tiene un límite de rendimiento para 15 atributos.

```
% Bucle para wrapper
    for c = 1:length(nombresAtributos)  %Generacion de combinaciones
        v = combnk([1:length(nombresAtributos)],c);

        for sz = 1:size(v,1)     % Recorrido de las combinaciones
            numConfig = numConfig + 1;
            Mm = M(:,v(sz,:));
            Mvm = Mv(:,v(sz,:));
            classM = M(:,end);
            classMv = Mv(:,end);
            atributosAMostrar = "";

            for att = 1:size(v(sz,:),2) % Cadena de atributos elegidos
                aux = v(sz,:);
                atributosAMostrar =
strcat(atributosAMostrar,nombresAtributos(aux(att)));
                atributosAMostrar = strcat(atributosAMostrar," ");
            end
            % Guardado de los atributos implicados en el entrenamiento
            atributos(avanzarWrapper) = atributosAMostrar;
```

El *wrapper* generará una combinación nueva en cada iteración del bucle, y seleccionará los datos para el conjunto de entrenamiento y el conjunto de validación a razón de los atributos, con sus vectores de etiquetas correspondientes.

- *trainingNet(X,Xnew,class,classNew,settings)*

Esta función realiza el entrenamiento y la validación del modelo, así como la obtención de las estadísticas de calidad necesarias. Recibe como parámetros lo siguiente:

- *X*: El conjunto de datos para el entrenamiento.
- *Xnew*: El conjunto de datos para la validación.
- *class*: Las etiquetas que determinan a qué clase pertenece cada fila de datos del conjunto de entrenamiento.
- *classNew*: Las etiquetas que determinan a qué clase pertenece cada fila de datos del conjunto de validación.
- *settings*: Conjunto de parámetros necesarios para entrenar la red en la configuración.

Dado que recibe todos los datos procesados, una vez acabada, devolverá las estadísticas obtenidas para que el bucle principal del *wrapper* las prepare para el guardado.

```
function [aucM,FPRM, TPRM, CUI_pos, CUI_neg, valores,
pred_class] = trainingNet(X,Xnew,class,classNew,settings)

    % Entrenamiento del modelo
    model = model_cpann(X,class,settings); % cpann

    % Validación - prediccion de etiquetas a nuevos ejemplares
    pred = pred_cpann(Xnew,model);

    % accuracy, precision, sensitivity, specificity
    valores = cpann_class_param(pred.class_pred', classNew);
    pred_class = pred.class_pred;

    % Matriz de confusion - para el cálculo de las estadísticas
    TP = valores.conf_mat(1,1);
    FP = valores.conf_mat(2,1);
    FN = valores.conf_mat(1,2);
    TN = valores.conf_mat(2,2);
    sp = TN/(FP + TN);
    sn = TP/(FN + TP);

    CUI_pos = getCUIPositive(sn,TP,FP);
    CUI_neg = getCUINegative(sp,TN,FN);

    % Cálculo AUC
    [sp,sn,auc,~,~,~,~] =
roc_curves(model.net.W_out,pred.top_map,classNew);
    aucM = auc(1);
    TPRM = sn;
    FPRM = 1 - sp;
end
```

- *showConfiguration(atributos, AUC, file, combination, settings, currentConfiguration, FPR,TPR, atributosAMostrar, ruta,CUI_pos,CUI_neg,valores)*

Esta es, probablemente, la función con mayor número de parámetros de todo el desarrollo. Se encarga de tabular los resultados y dirigir la información necesaria para guardarlos y graficarlos. Los parámetros que recibe son:

- *atributos*: Un vector con las combinaciones de todos los atributos.
- *AUC*, *FPR*, *TPR*, *CUI_pos* y *CUI_neg*: Los vectores con las estadísticas del modelo de cada combinación.
- *combination*: Cell array[24] de todas las configuraciones que se quieren probar.
- *settings*: Conjunto de parámetros utilizado para identificar la configuración tabulada.
- *currentConfiguration*: Configuración de red actual. Utilizada para identificar la tabla.

[24] Tipo de datos que indexa los datos en celdas. Puede contener cualquier tipo de dato (MatLab 2020).

- *AtributosAMostrar*: Atributos del conjunto de datos. Utilizados para identificar la tabla.
- *Ruta*: Directorio base de guardado de los resultados.
- *file*: Nombre del fichero que contiene los datos. Utilizado para identificar la tabla.
- *Valores*: Valores finales de la red. Utilizado para análisis posterior.

```
% Mostrado del AUC
  fprintf("\n-----------Tabla de AUCs con orden por AUC de trapz-------
----");
  AUCs =
sortrows(table(atributos,AUCM,CUI_pos,CUI_neg,accuracy,sensitivity,spec
ificity,error),2,'descend');
  AUCs.Configuracion = config;
  AUCs.DatasetsUsados = dataset;

  fichero = strtok(file,'P');

  % Guardado de la tabla
  config =
sprintf('%d_%d_%s_%s_%s_%s_%d_%f_%s_%d_%d_%d',settings.nsize,settings.e
pochs,settings.topol, settings.bound,
settings.training,settings.init,settings.a_max,settings.a_min,settings.
scaling,settings.absolute_range,settings.show_bar,settings.ass_meth);
  file = sprintf('%s-%s-%s',config,fichero,atributosAMostrar);
  directory = [ruta '\tablas\Final\'];
  if exist(directory,'dir') == 0  % Comprobamos si existe el
directorio, si no lo creamos
     mkdir(directory);
  end
  save(strcat([ruta '\tablas\Final\' file],'.mat'),'AUCs');


  % Mostrado de la curva ROC
  currentConfiguration =
GetNConfig(directory,config,'diccionario.mat');
  bestAUC = AUCs.(2)(1);

  % Guardado de FPR-TPR para mostrado sin ejecución
saveFPR_TPR(directory,'FPR_TPR.mat',currentConfiguration,FPR,TPR,bestAU
C);
  mostradoTest(FPR,TPR,currentConfiguration,ruta,bestAUC,0);
  hold off;
  saveas(gcf,strcat(ruta, '\graficas\Final\', file,'.fig'));
```

- *mostradoTest(FPR,TPR,config,ruta,auc,sameFigure,linespec)*

  La función general para graficar la curva ROC. Recibe como parámetros lo siguiente:

  - *FPR* y *TPR*: La X y la Y necesarias para dibujar la curva.
  - *config*: El nombre identificativo de la configuración a graficar.
  - *ruta*: Directorio base donde se guardará la gráfica resultante.

- *auc*: El mejor resultado de AUC obtenido por el clasificador entre todas sus combinaciones.
- *sameFigure*: Un parámetro para determinar si se crea una figura nueva o se sigue usando la existente.
- *Linespec*: Parámetro optativo que permite cambiar el color, el trazado y los símbolos de los puntos de la curva.

Esta función está preparada para permitir el graficado de más de una curva ROC en una misma figura. Cuando se ha creado la línea deseada, modificará la leyenda para añadir la nueva configuración con su respectivo valor de AUC.

```
plot(FPR(1,:),TPR(1,:),linespec, 'LineWidth',2),   %AUC

    set(gcf,'defaultLegendAutoUpdate','off');
    title('ROC');
    if isempty(nConfig)
        currentConfig = sprintf('%s - AUC:%0.4f',config,auc);
    else
        currentConfig = sprintf('Config%d - AUC:%0.4f',nConfig,auc);
    end

    if isempty(findobj(gcf, 'Type','Legend')) == 1
        x = linspace(0,1);
        y = linspace(0,1);
        plot(x,y,'--r');    % Clasificador random - diagonal
        nombres = {currentConfig,'Random'};

        legend(nombres,'Location','best');

    else
        legappend(currentConfig);
    end

    xlabel("FPR = 1 - Specificity");
    ylabel("TPR = Sensitivity");
```

- *weightedMajorityVoting(X,Xnew,class,classNew,settings,weightPercentages,ruta,useDifferentAtributes,atributesMatrix)*

Función que implementa el método de WMV para un problema binario. El método SMV es un subcaso de este, en el cual los pesos especificados para cada clasificador son todos el mismo. Recibe los siguientes parámetros:

- *X* y *Xnew*: Conjuntos de datos de entrenamiento y validación respectivamente.
- *Class* y *classNew*: Etiquetas de clase de los conjuntos de entrenamiento y de validación.
- *Settings*: Vector de configuraciones para los clasificadores individuales.
- *weightPercentages*: Vector de pesos para determinar qué clasificadores influyen más o menos en el ensemble.
- `ruta`: Directorio base para guardar los resultados.
- *useDifferentAtributes*: Parámetro para modificar el modo de selección de atributos. En este proyecto se mantendrá con el valor de 2, de manera que

siempre usará para cada clasificador los atributos determinados desde un fichero.
- *atributesMatrix*: Atributos de entrenamiento especificados a través de un fichero.

Funciones de apoyo desarrolladas

Aparte de los programas principales que definen el flujo básico del sistema de detección, se han creado una serie de funciones de apoyo con el objetivo de facilitar el análisis posterior de los resultados. Estas se encuentran definidas en sus propios ficheros, siguiendo la regla de MatLab en la que la función principal debe tener el mismo nombre que el fichero.

Este repertorio de funciones adicionales se explica a continuación:

- *getEstadisticaDescriptiva(data, atributos, directory)*

Obtiene la estadística descriptiva de un conjunto de datos representados en Tabla 4 y Tabla 5. Recibe la totalidad de los datos del conjunto, los atributos y el directorio base para guardar los resultados.

- *saveFPR_TPR(directorio,nombreFichero,config,FPR,TPR,bestAUC)*

Guarda los valores de FPR, TPR y AUC para la mejor combinación de la configuración determinada por parámetro. Esto facilita el graficado de cualquier configuración una vez acabado el entrenamiento.

- *getCUINegative(specificity, TN,FN) / getCUIPositive(sensitivity, TP,FP)*

Obtiene el valor de CUI positivo y de CUI negativo a partir de las mediciones del modelo.

- *GetNConfig*(directorio, config, nombreFichero)

A partir de una configuración, se obtiene el nombre de la entrada guardada. Es especialmente útil para obtener la relación de las configuraciones con sus valores FPR, TPR y AUC guardados.

- *saveValores(ruta, fileName, settings, atributos, valores, pred_class)*

Guarda en un fichero aparte todas las mediciones del modelo en caso de querer hacer una consulta posterior de información no tabulada.

- *PlotAllAUCTogether(*configs)

Recibe un cell array con todas las configuraciones que se desea graficar y hace uso de *mostradoTest* para mantener la información de cada curva ROC. Esta función existe para crear una figura de conjuntos tras el análisis de las combinaciones, obteniéndose una gráfica como la Figura 24.

Capítulo 3: Resultados y discusión

En este último capítulo se discutirán los resultados obtenidos para cada sistema de detección.

Resultados del sistema de detección: redes neuronales híbridas

Para realizar un estudio controlado, se ha aplicado una estrategia de pruebas jerárquicas. De esta manera, se empieza variando un parámetro con todas las posibilidades y, a medida que se van realizando nuevas pruebas, se va reduciendo el conjunto de configuraciones. Esto permite acotar la búsqueda de la mejor configuración a un tiempo más corto.

Dado que la variación es de cuatro parámetros y siempre es necesario que tengan un valor, se han aplicado de manera distinta según el efecto que causan en la red. Así pues, el tamaño y las épocas se variarán continuamente, mientras que el entrenamiento y los límites del mapa permanecerán como las referencias.

A continuación, se presentarán los resultados de cada conjunto de datos (A y luego B), y las pruebas realizadas en cada caso. Para cada prueba, primero se mostrará una gráfica con los cambios del AUC de las mejores combinaciones obtenidas frente a las épocas de cada tamaño de red. Seguidamente, se mostrará una tabla con los resultados de las configuraciones según diferentes métricas de calidad, y se acabará con una gráfica de las curvas ROC de las configuraciones ganadoras.

Conjunto A

El conjunto A de datos se comenzó a probar con los valores:

- Tamaño: 4, 8, 10, 12 y 14.
- Épocas: 10, 100, 200, 500 y 1000.
- Límite: Toroidal.
- Entrenamiento: *Batch*.

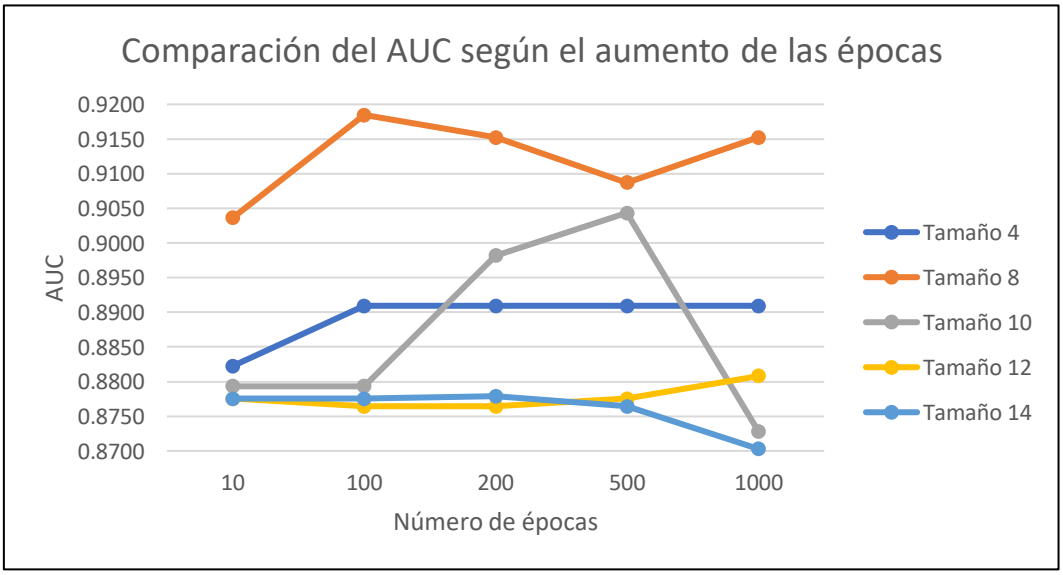

Figura 21. Evolución del AUC para configuraciones de límite toroidal y entrenamiento batch.

Como se puede ver en la Figura 21, la mejor configuración con diferencia es la de tamaño 8, pues tiene un AUC por encima del 0,90. En general, se nota una estabilidad[25] en la red, que se ve perturbada únicamente en la configuración de tamaño 10 al variar casi tres puntos (0,87 a 0,90) enteros.

Para facilitar la lectura de aquí en adelante, se les dará un nombre a las configuraciones que se abreviará a "ConfigX" en casos posteriores. En la Tabla 8 se muestra la asignación para las configuraciones que comparten el par *toroidal-batch*:

Tabla 8. Correspondencia de nombres y tipos de configuraciones para la primera prueba del conjunto A.

Tamaño	Épocas	Nombre
4	100	Configuración 2
8	100	Configuración 7
10	500	Configuración 14
12	1000	Configuración 20
14	500	Configuración 24

Por su parte, estas configuraciones tienen un buen CUI para la detección de casos positivos en Deterioro Cognitivo Leve (DCL), aunque moderado para los casos negativos. La precisión es bastante alta, ya que ronda el 80% de los casos como se puede apreciar en la Tabla 9.

En cuanto a los atributos ganadores, las configuraciones cuentan con al menos dos pruebas específicas de la Enfermedad de Alzheimer, en especial MMSE y FAQ. También aparece la edad en una ocasión, lo que la convierte en un factor importante.

Tabla 9. Tabla de resultados para las configuraciones ganadoras de la primera prueba del conjunto A.

Configuración	Atributos	AUC	CUI+	CUI-	Accuracy	Sensitivity	Specificity	Error
Config2	MMSE FAQ	0,8909	0,7458	0,5975	0,8289	0,8913	0,7333	0,1877
Config7	Age MMSE FAQ GDS	0,9185	0,7516	0,5400	0,8158	0,9565	0,6000	0,2217
Config14	MMSE FAQ	0,9043	0,6972	0,4587	0,7763	0,9130	0,5667	0,2601
Config20	MMSE FAQ	0,8808	0,7028	0,5014	0,7895	0,8913	0,6333	0,2377
Config24	FAQ GDS	0,8764	0,6824	0,5378	0,7895	0,8261	0,7333	0,2203

Finalmente, en las curvas ROC se puede apreciar en la Figura 22 cómo varían los falsos positivos (FPR, del inglés *False Positive Rate*) frente a los verdaderos positivos (TPR, del inglés *True Positive Rate*), resultado de las decisiones del sistema para cada configuración. En esta se puede ver cómo Config7 se encuentra por encima de las demás, más cerca del ideal buscado de FPR a 0 y TPR a 1.

Config7 solo es sobrepasada puntualmente por Config14 y Config24, lo que indica que, para esos rangos, sendas configuraciones son mejores.

[25] La inestabilidad de la red, definida como una falta de equilibrio (Real Academia Española 2005), se tendrá en cuenta a lo largo de este apartado como la variación brusca entre valores máximos y mínimos del AUC obtenido a lo largo de toda la prueba para una determinada configuración. Esta variación puede ser mayor o menor, pero se entenderá como "preocupante" si la diferencia es superior a tres puntos.

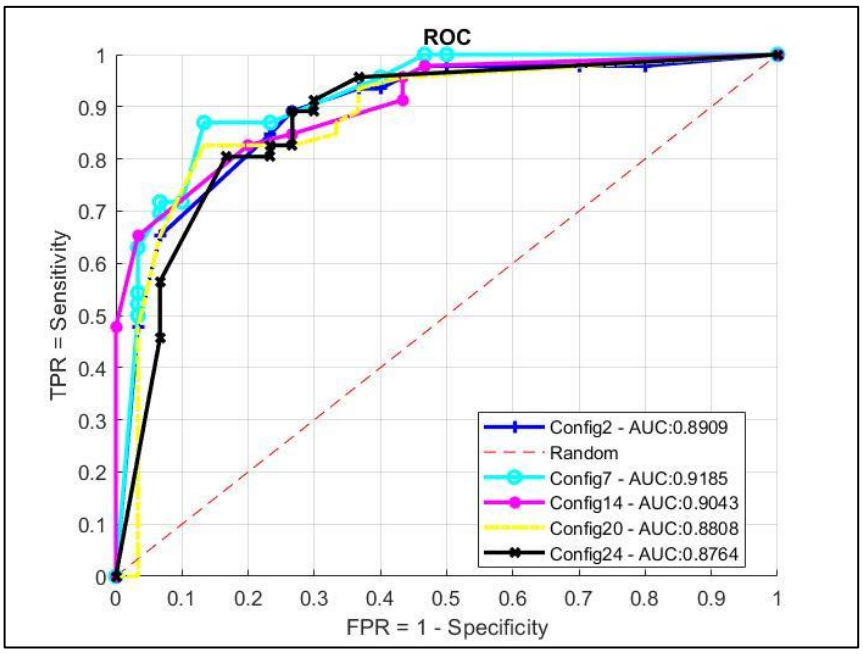

Figura 22. Comparativa de curvas ROC de las configuraciones ganadoras de la primera prueba del conjunto A.

A continuación, se ha variado el límite del mapa, quedando así las configuraciones:

- Tamaño: 4, 8, 10, 12 y 14.
- Épocas: 10, 100, 200, 500 y 1000.
- Límite: Normal.
- Entrenamiento: *Batch*.

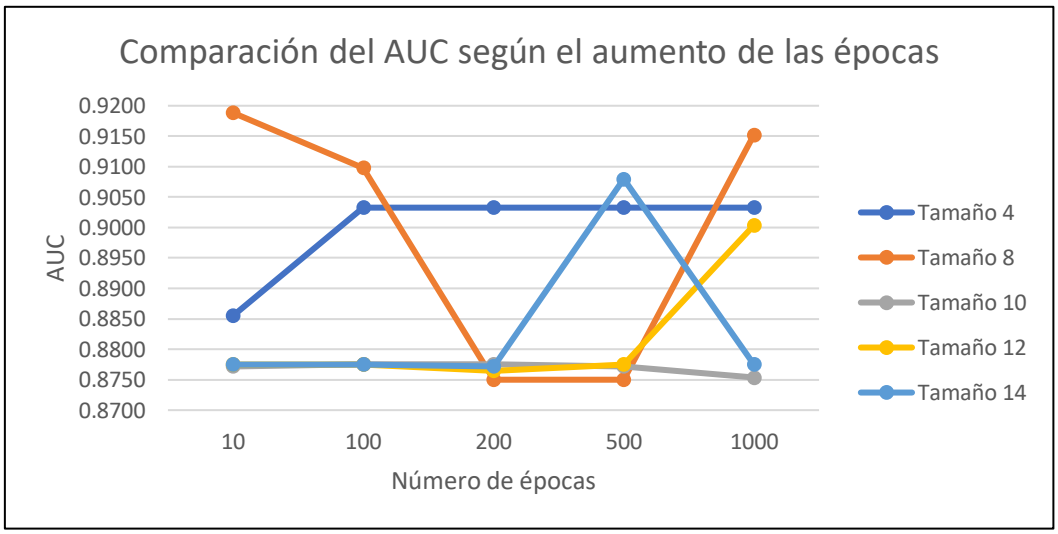

Figura 23. Evolución del AUC para configuraciones de límite normal y entrenamiento batch.

Para estas pruebas, se ha detectado una mayor inestabilidad en comparación con el sistema anterior. Como se puede ver en la Figura 23, la mejor configuración de tamaño 8 ha producido un valle cuya diferencia entre el mejor y el peor valor es de casi cinco puntos (de 0,87

a 0,92). De esta manera, con un límite normal se ha convertido en una configuración descartable.

Por otro lado, la configuración de tamaño 14 tiene un pico que también la descarta, puesto que es de tres punto (0,87 a 0,90). Además, se aprecia cómo la configuración de tamaño 12 ha mejorado su comportamiento para aumentar los valores.

Para facilitar la lectura de aquí en adelante, en la Tabla 10 se muestra la asignación para las configuraciones que comparten el par *normal-batch*:

Tabla 10. Correspondencia de nombres y tipos de configuraciones para la segunda prueba del conjunto A.

Tamaño	Épocas	Nombre
4	100	Configuración 27
8	100	Configuración 31
10	500	Configuración 37
12	1000	Configuración 45
14	500	Configuración 49

Como se muestra en la Tabla 11, el AUC ha mejorado en casi todos los casos, pero el resto de los resultados ha disminuido en general para las mejores configuraciones. En cuanto a los atributos, se encuentran solo aquellos tests específicos para la detección de DCL, convirtiéndose en pares de MMSE y FAQ o FAQ y GDS.

Tabla 11. Tabla de resultados para las configuraciones ganadoras de la segunda prueba del conjunto A.

Configuración	Atributos	AUC	CUI+	CUI-	Accuracy	Sensitivity	Specificity	Error
Config27	MMSE FAQ	0,9033	0,7309	0,5654	0,8158	0,8913	0,7000	0,2043
Config31	MMSE FAQ	0,9188	0,7613	0,6298	0,8421	0,8913	0,7667	0,1710
Config37	FAQ GDS	0,8775	0,7035	0,5563	0,8026	0,8478	0,7333	0,2094
Config45	MMSE FAQ	0,9004	0,6895	0,4696	0,7763	0,8913	0,6000	0,2543
Config49	MMSE FAQ	0,9080	0,6895	0,4696	0,7763	0,8913	0,6000	0,2543

Finalmente, en la Figura 24 se pueden ver las curvas ROC. Estas configuraciones coinciden numerosas veces. Config31 es la mejor según su valor de AUC, pero la diferencia no es demasiado con las demás. Esto sugiere que las configuraciones ganadoras tienen un comportamiento similar entre sí y podrían aportar un resultado igual de satisfactorio con menor coste.

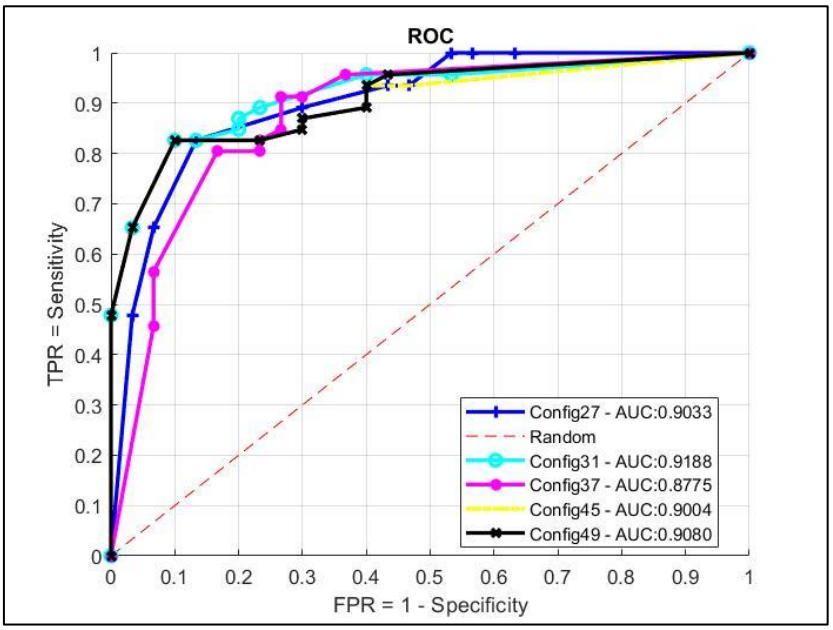

Figura 24. Comparativa de curvas ROC de las configuraciones ganadoras de la segunda prueba del conjunto A.

Manteniendo el parámetro de límite en toroidal, se ha realizado el cambio del entrenamiento y también se han fijado las épocas en la mejor configuración observada. En esta ocasión, además, se ha decidido usar casos extremos de épocas y aumentarlo un poco:

- Tamaño: 8.
- Épocas: 100, 1000 y 5000.
- Límite: Toroidal.
- Entrenamiento: *Sequential*.

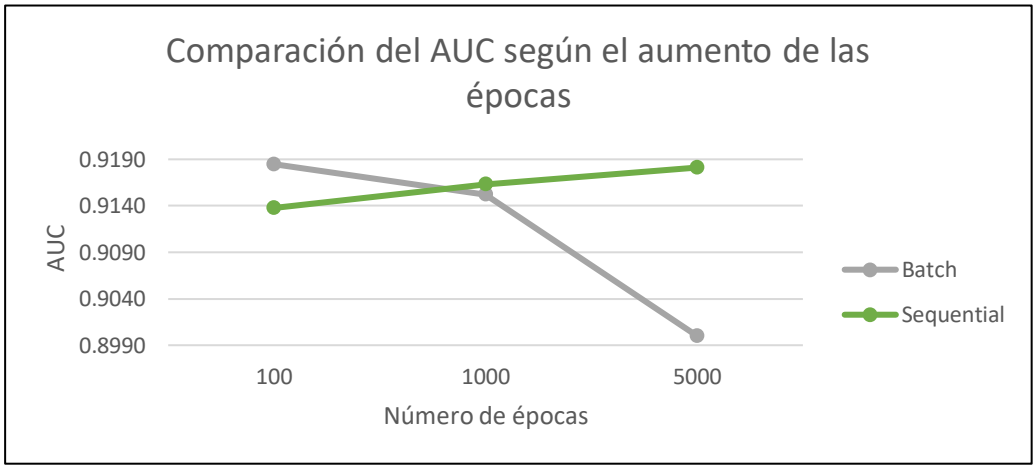

Figura 25. Evolución del AUC para configuraciones de entrenamiento tipo sequential y tipo batch.

Al realizar el cambio de entrenamiento, observando la Figura 25 se descubre que *batch* pierde eficacia a mayor número de épocas a pesar de que, inicialmente, produce mejores resultados. Por su parte, *sequential* mejora muy ligeramente el AUC.

Para facilitar la lectura de aquí en adelante, en la Tabla 12 se muestra la asignación para las configuraciones que comparten el par *normal-batch*:

Tabla 12. Correspondencia de nombres y tipos de configuraciones para la tercera prueba del conjunto A

Entrenamiento	Épocas	Nombre
Batch	100	Configuración 7
Batch	1000	Configuración 10
Batch	5000	Configuración 51
Sequential	100	Configuración 52
Sequential	1000	Configuración 53
Sequential	5000	Configuración 54

Además, la calidad del sistema con *batch* se pierde ligeramente a medida que las épocas aumentan, con una diferencia que ronda un punto aproximadamente, por ejemplo 0,90 a 0,91 en AUC para Config51 y Config54 respectivamente. También se puede apreciar un menor error con una configuración que utiliza entrenamiento *sequential*, visto en la Tabla 13.

Configuración	Atributos	AUC	CUI+	CUI-	Accuracy	Sensitivity	Specificity	Error
Config7	Age MMSE FAQ GDS	0,9185	0,7516	0,5400	0,8158	0,9565	0,6000	0,2217
Config10	MMSE FAQ	0,9152	0,6895	0,4696	0,7763	0,8913	0,6000	0,2543
Config51	MMSE FAQ	0,9000	0,6895	0,4696	0,7763	0,8913	0,6000	0,2543
Config52	MMSE FAQ	0,9138	0,7246	0,5762	0,8158	0,8696	0,7333	0,1986
Config53	MMSE FAQ	0,9163	0,7375	0,5556	0,8158	0,9130	0,6667	0,2101
Config54	MMSE FAQ	0,9181	0,7098	0,5444	0,8026	0,8696	0,7000	0,2152

Tabla 13. Tabla de resultados para las configuraciones ganadoras de la tercera prueba del conjunto A

No obstante, el comportamiento de los sistemas es bastante parecido, como se puede comprobar en la Figura 26, donde se encuentra que prácticamente todos coinciden hasta casi el final de la curva ROC.

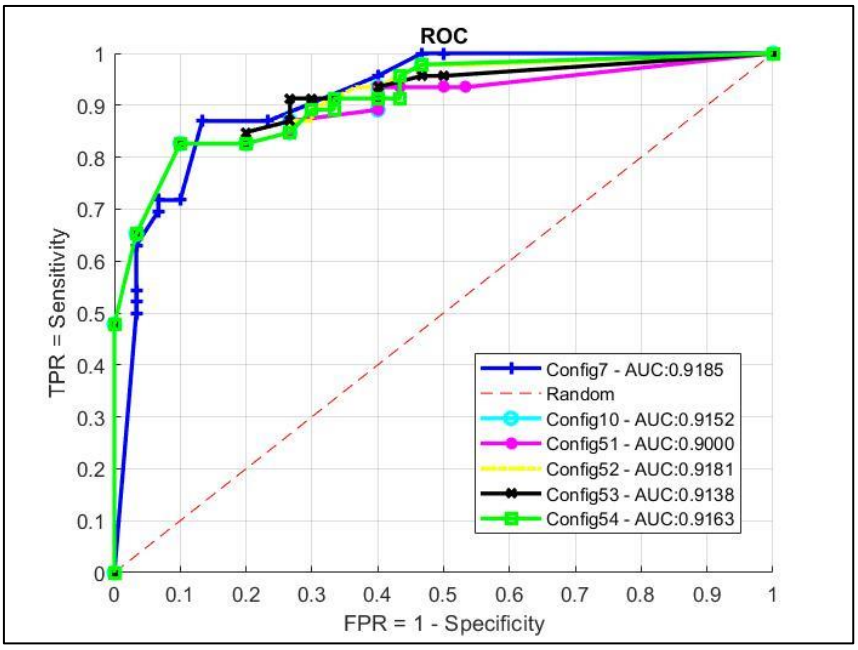

Figura 26. Comparativa de curvas ROC de las configuraciones ganadoras de la tercera prueba del conjunto A

Dado que el tipo de entrenamiento puede aumentar las iteraciones realizadas en cada época, *batch* y *sequential* se distinguen principalmente por el tiempo que tardan en dar un resultado. Debido a esto, este tipo de medida es necesaria para decidir la idoneidad de cada uno.

En la Tabla 14 se encuentran los tiempos que ha tardado cada sistema en acabar su ejecución. Finalmente, se calculan los ratios de diferencia entre los distintos entrenamientos.

Tabla 14. Comparación de los tiempos en segundos entre cada par de configuraciones.

Tamaño	Batch	Sequential	Ratios
100	14,69	25,54	1,74
1000	87,93	184,13	2,09
5000	453,14	862,53	1,90

El tiempo de un entrenamiento *sequential* es casi el doble que el de un entrenamiento *batch* de manera generalizada. Esto implica que, para las mismas configuraciones, se tendría que esperar el doble de tiempo. Teniendo en cuenta que el AUC de las configuraciones con entrenamiento *batch* disminuye de manera ligera con respecto al otro componente de esta prueba, la ganancia de *sequential* frente a la espera para recibir una respuesta se convierte en desechable.

Conjunto B

El conjunto B de datos es un estudio nuevo dado que contiene distintos tipos de datos. No obstante, se han limitado casos gracias a la demostración de que el entrenamiento *batch* y el límite toroidal son mejores. Los nuevos valores utilizados han sido:

- Tamaño: 4, 8, 10 ,12 ,14 y 16.
- Épocas: 10, 100, 200, 500 y 1000.
- Límite: Toroidal.
- Entrenamiento: *Batch*.

Las configuraciones para estos datos han presentado una mejor estabilidad, no variando en más de dos puntos excepto en el caso de tamaño 8, que ha superado los cuatro puntos (de 0,82 a 0,87). Como se ve en la Figura 27, las configuraciones en este caso se han mantenido o disminuido con el aumento de las épocas.

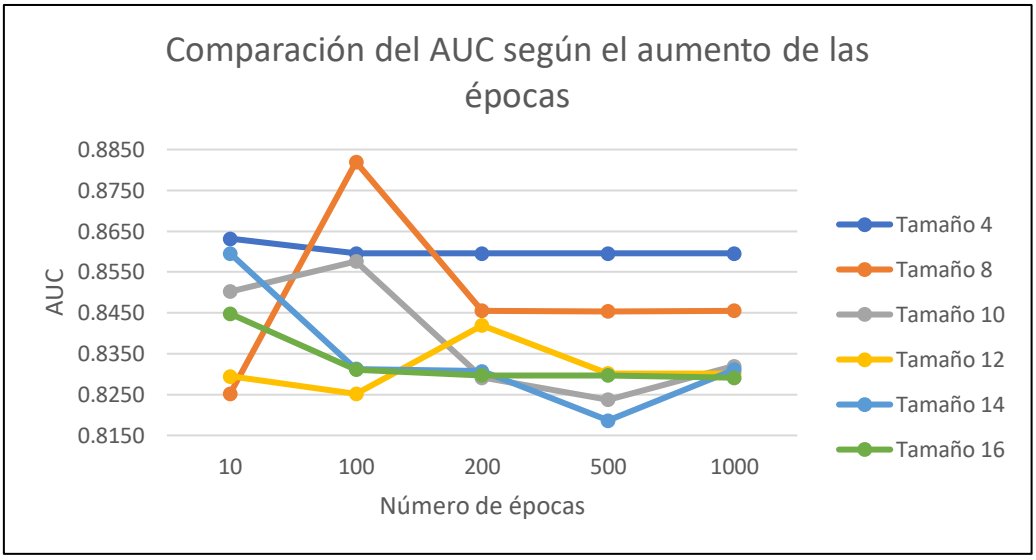

Figura 27. Evolución del AUC del conjunto B para configuraciones de límite toroidal y entrenamiento batch.

Para facilitar la lectura de aquí en adelante, en la Tabla 15 se muestra la asignación para las configuraciones que comparten el par *normal-batch*:

Tabla 15. Correspondencia de nombres y tipos de configuraciones para la prueba del conjunto B.

Tamaño	Épocas	Nombre
4	10	Configuración 1
8	100	Configuración 7
10	100	Configuración 12
12	200	Configuración 18
14	10	Configuración 21
16	10	Configuración 55

Las configuraciones ganadoras han sido realmente estables, presentando siempre el trío de edad, MOCA y NPITOTAL, junto con FAQ en la mayoría de los resultados. A pesar de que varían bastante según las épocas, en la Tabla 16 se puede apreciar cómo tienen una media de

AUC de 0,85. Además, el CUI positivo se mantiene moderado y el CUI negativo es bueno, lo cual implica que, a pesar de no tener la mejor evolución, podrían tratarse de sistemas útiles para la detección de DCL.

Tabla 16. Tabla de resultados para las configuraciones ganadoras de la prueba del conjunto B.

Configuración	Atributos	AUCM	CUI+	CUI-	Accuracy	Sensitivity	Specificity	Error
Config1	AGE MOCA NPITOTAL	0,8632	0,5269	0,7147	0,8018	0,7632	0,8219	0,2075
Config7	AGE MOCA FAQ NPITOTAL	0,8819	0,5531	0,6614	0,7748	0,8947	0,7123	0,1965
Config12	AGE MOCA FAQ NPITOTAL	0,8576	0,5757	0,6735	0,7838	0,9211	0,7123	0,1833
Config18	AGE MOCA FAQ NPITOTAL	0,8419	0,5432	0,6478	0,7658	0,8947	0,6986	0,2033
Config21	AGE MOCA FAQ NPITOTAL	0,8596	0,5558	0,6462	0,7658	0,9211	0,6849	0,1970
Config55	AGE MOCA FAQ NPITOTAL	0,8448	0,5156	0,6070	0,7387	0,8947	0,6575	0,2239

Estudiando las curvas ROC de estos modelos, aunque Config7 es el mejor sistema obtenido en cuanto a su AUC, a lo largo de la representación se ve que esta distinción es simplemente inicial; luego se encuentran otras configuraciones con mejores respuesta.

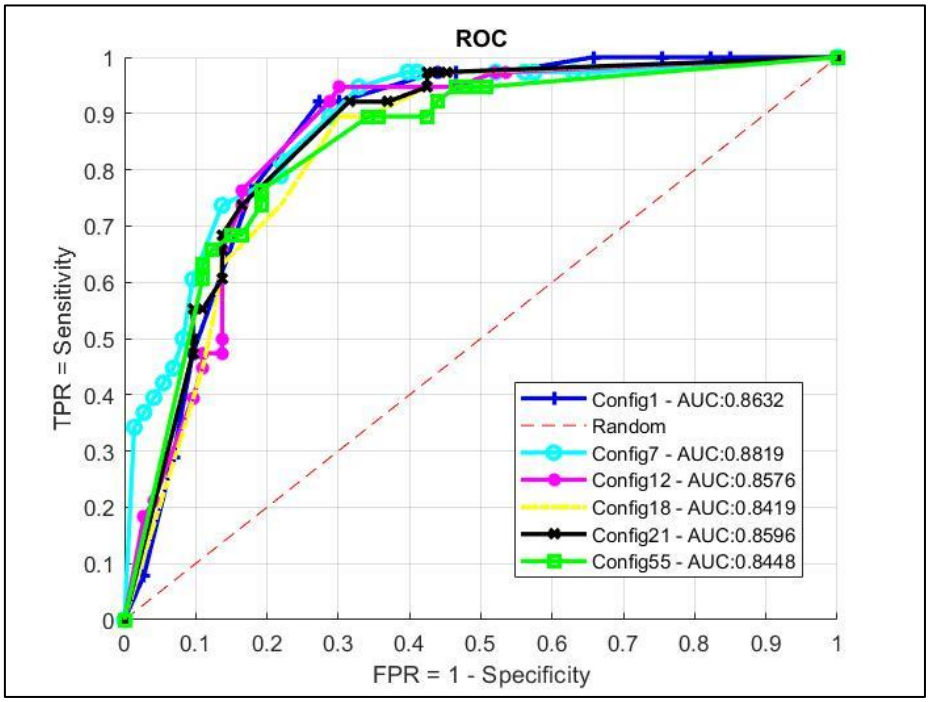

Figura 28. Comparativa de curvas ROC de las configuraciones ganadoras de la prueba del conjunto B.

En la Figura 28 se puede apreciar cómo Config1 y Config12 se encuentran igual de bien posicionados para determinadas elecciones, por lo que fallan menos en esos rangos. Esto corrobora los valores obtenidos en *accuracy* para estos sistemas, que mejoran los de Config7.

Pruebas de tiempo

Otro factor muy importante para medir la utilidad del sistema es el tiempo que tarda en realizar sus decisiones. Puesto que esta herramienta es para la detección de una enfermedad, este cobra un interés extra que podría significar la diferencia entre un diagnóstico tarde o a tiempo.

Aparte de las estadísticas de calidad, se ha medido el tiempo de entrenamiento de cada clasificador realizado por el *wrapper*. Estas pruebas son interesantes sobre todo para la comprobación del rendimiento del sistema, puesto que se pueden tener modelos con resultados aproximadamente iguales cuyo coste en tiempo sea completamente distinto.

En general, a mayor tamaño y épocas, más tardará el sistema en dar los resultados. Este aumento, además, es proporcional a ambos, de una manera prácticamente lineal. En la Figura 29 correspondiente a los clasificadores con entrenamiento *batch* y límite toroidal del conjunto A, es posible apreciar cómo se da este comportamiento cuando el tiempo aumenta junto con las épocas. Además, incrementar el tamaño de la red hace que el tiempo también aumente en consecuencia con respecto a clasificadores de menor tamaño.

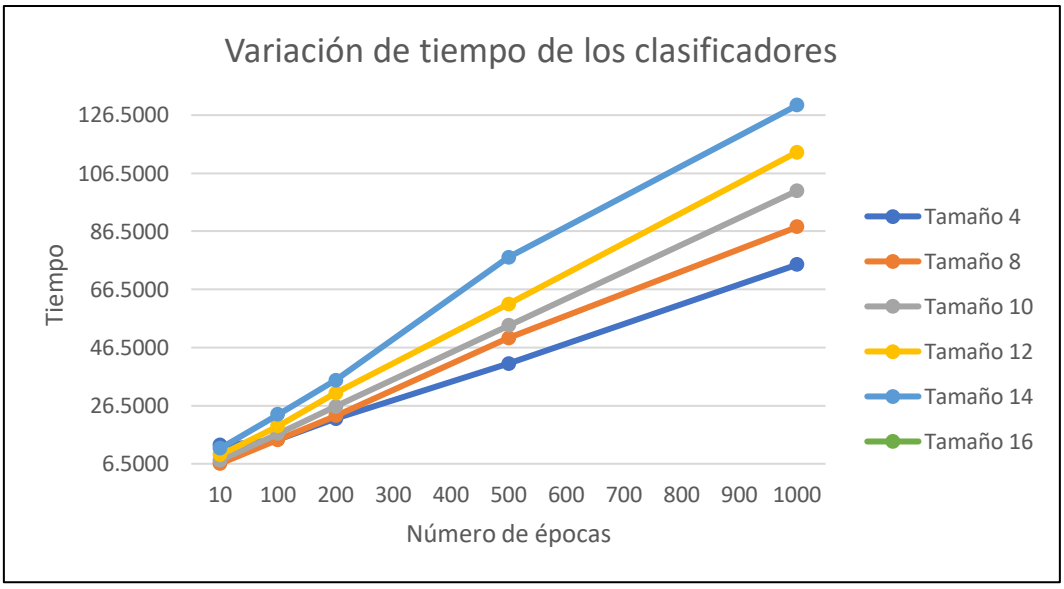

Figura 29. Ejemplo del aumento del tiempo para los clasificadores de la primera prueba del conjunto A.

Mirando con mayor detalle a los valores de los clasificadores, se puede ver en la Tabla 17 la progresión del aumento del tiempo, que es de aproximadamente el doble en todos los casos. Además, el clasificador de tamaño 8 y el de tamaño 12 tienen tiempos menores que los demás para 10 épocas. Además, el de tamaño 8 no consume demasiado tiempo en general, lo que corrobora que el clasificador 8 de esta prueba es mejor tanto en resultados como en tiempo.

Tabla 17. Tiempos de la primera prueba del conjunto A.

Tamaño	10	100	200	500	1000
4	12,80	14,81	21,99	40,91	74,98
8	6,55	14,69	23,02	49,78	87,93
10	7,68	16,76	26,21	53,99	100,39
12	9,50	19,50	30,75	61,44	113,70
14	11,77	23,43	35,16	77,52	129,80

En la Tabla 18 se encuentran los tiempos de los clasificadores con entrenamiento *batch* y límite normal del conjunto A. En esta se aprecia una estabilidad mayor con respecto a la tabla anterior, siguiendo la regla esperada. A mayores épocas, no obstante, se nota que la red es un poco más lenta que con el límite toroidal, aunque no lo suficiente como para desechar estas combinaciones.

Tabla 18. Tiempos de la segunda prueba del conjunto A.

Tamaño	10	100	200	500	1000
4	6,61	14,49	22,26	45,15	83,87
8	7,21	16,22	25,83	52,17	96,36
10	8,31	18,10	28,59	58,78	107,86
12	10,25	20,54	31,80	64,55	119,48
14	12,66	22,98	35,08	71,60	130,67

Finalmente, para la prueba del conjunto B se muestra la Tabla 19. Para los casos más pequeños, épocas y tamaños más pequeños, estos clasificadores son capaces de dar una respuesta con mayor rapidez con respecto a los de la primera prueba del conjunto A. No obstante, a medida que van aumentando, este se dispara hasta casi doblar el tiempo de ambas pruebas del conjunto A. Esto sugiere, aparejado con los datos del apartado anterior, que los datos de estas pruebas cognitivas son generalmente más costosos.

Tabla 19. Tiempos de la prueba del conjunto B.

Tamaño	10	100	200	500	1000
4	9,02	15,79	23,38	47,89	88,17
8	7,42	17,56	27,15	56,44	106,03
10	8,73	20,34	32,06	67,67	127,84
12	10,88	23,03	38,74	83,95	149,68
14	12,36	24,16	37,13	77,10	138,98
16	14,34	27,54	41,91	84,77	157,44

Resultados del sistema de detección: ensembles neuronales

En este caso, se ha elegido hacer un ensemble neuronal compuesto por varias redes de tipo Counterpropagation.

Como estrategia de diversidad se ha analizado la correlación entre los clasificadores. Para ello, se ha realizado un análisis de los errores de los clasificadores de una configuración que se ha determinado como buena en el estudio anterior, y variado la entrada de criterios clínicos que recibe cada uno según la baja correlación entre estos. Así, cada clasificador tiene la seguridad de errar en un espacio distinto.

Dado que con este proceso se dividen los atributos entre los distintos clasificadores que conforman el ensemble, se ha elegido tener cinco sistemas, permitiendo la repetición de atributos sin que se dé la misma combinación que otro previamente entrenado.

Por otra parte, para la combinación de salidas se ha escogido una estrategia a nivel de etiquetas de clase basada en votación. Estos son la votación por mayoría simple (SMV, del inglés *simple majority voting*) y la votación por mayoría ponderada (WMV, del inglés *weighted majority voting*).

Durante el desarrollo de esta nueva prueba, se ha detectado que el cálculo de la curva ROC obtiene los mismos resultados para un método SMV como para un método WMV. La estrategia utilizada para calcular la curva ROC (y obtener el AUC consecuentemente) en el ensemble neuronal es la del máximo de la salida aportada por los clasificadores individuales que lo componen. Ello conlleva a que la estrategia de combinación no influye en el cálculo de la curva ROC y, por tanto, es una medida cuya utilidad se pierde para las comparaciones con los mismos ensembles.

A pesar de ello, es una estrategia válida y sí permite una comparación directa con un sistema compuesto por un clasificador individual. Debido a ello, se aportan ambas curvas ROC en sus apartados respectivos independientemente de que se mantengan los resultados.

Conjunto A

Para el primer conjunto de datos con una estrategia de combinación SMV se han utilizado las siguientes combinaciones:

Configuración general: 12 100 hexagonal toroidal batch eigen 1 0.1 auto 0 1 1

- Clasificador 1 (Azul): Edad, MMSE y GDS.
- Clasificador 2 (Rojo): Edad, MMSE, FAQ y GDS.
- Clasificador 3 (Verde): FAQ, GDS y años de educación.
- Clasificador 4 (Celeste): MMSE y años de educación.
- Clasificador 5 (Rosa): FAQ y años de educación.

Tabla 20. Resultados del primer ensemble (SMV).

Clasificador	AUC	Error	CUI+	CUI-
Clasificador 1 (Azul)	0.6645	0.3080	0.5506	0.4040
Clasificador 2 (Rojo)	0.8293	0.2486	0.6820	0.4813
Clasificador 3 (Verde)	0.7446	0.2536	0.6540	0.4762
Clasificador 4 (Celeste)	0.6957	0.3486	0.6540	0.4762
Clasificador 5 (Rosa)	0.6855	0.2768	0.6540	0.4762
Ensemble	0.8391	0.2210	0.7165	0.5333

Como se puede apreciar en la Tabla 20, se han combinado cinco clasificadores con un error no muy grande en general que ronda un 20-30%, inicialmente con un CUI bueno en general para valores positivos, aunque moderado en negativos.

El ensemble resultante obtiene un AUC de 0.8391, disminuyendo los errores de los sistemas que lo componen y manteniendo un índice de utilidad clínica bueno positivo, pero moderado negativo.

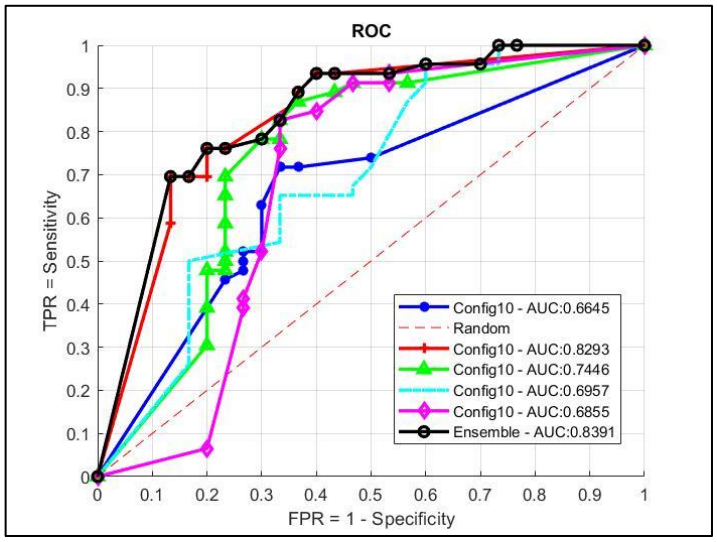

Figura 30. Curva ROC del ensemble y los clasificadores que lo componen (SMV).

A la hora del graficado multipunto, se recalcula el AUC de los sistemas para hacer una aproximación con más puntos. Como se puede apreciar en la Figura 30, este se ajusta a los puntos máximos, que coinciden en su mayoría con el Clasificador 2 (rojo).

Continuando con los mismos clasificadores, se ha modificado la estrategia de combinación a la WMV. En este caso, se han dado los pesos teniendo en cuenta el desempeño de los clasificadores individuales en la prueba anterior, obteniéndose los resultados de la Tabla 21.

Tabla 21. Resultados obtenidos del segundo ensemble (WMV).

Clasificador	Pesos	AUC	Error	CUI+	CUI-
Clasificador 1 (Azul)	0.1	0.6645	0.3080	0.5506	0.4040
Clasificador 2 (Rojo)	0.3	0.8293	0.2486	0.6820	0.4813
Clasificador 3 (Verde)	0.2	0.7446	0.2536	0.6540	0.4762
Clasificador 4 (Celeste)	0.2	0.6957	0.3486	0.6540	0.4762
Clasificador 5 (Rosa)	0.2	0.6855	0.2768	0.6540	0.4762
Ensemble	-	0.8391	0.2159	0.7444	0.5470

En este caso, el peso no afecta al entrenamiento de los clasificadores individuales. Esto se puede ver en cómo se mantienen los valores, mientras que el del ensemble mantiene su resultado en el AUC.

En cuanto al error, en el caso del voto mayoritario el ensemble disminuye notoriamente con respecto a la estrategia de combinación anterior puesto que ahora hay clasificadores con mayor influencia que otros. El índice de utilidad clínica, por su parte, ha mejorado bastante.

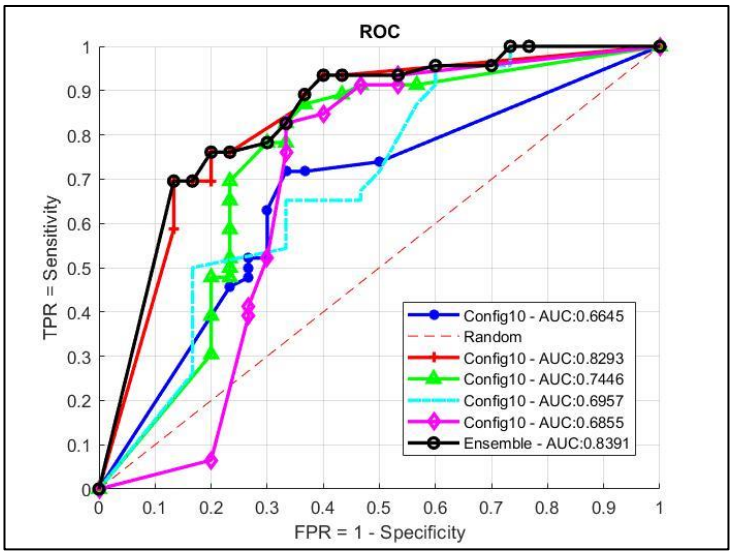

Figura 31. Curva ROC del ensemble y los clasificadores que lo componen (WMV).

Por la parte del graficado, igual que en el caso anterior, se modifica el AUC para la aproximación. El cálculo para este caso es el mismo y se mantiene puesto que el graficado de la Figura 31 depende los clasificadores individuales, que no modifican su comportamiento.

Conjunto B

Para el segundo conjunto de datos con una estrategia de combinación SMV se han utilizado las siguientes combinaciones:

Configuración general: 12 100 hexagonal toroidal batch eigen 1 0.1 auto 0 1 1

- Clasificador 1 (Azul): FAQ, NPITOTAL.
- Clasificador 2 (Rojo): Edad y MOCA.
- Clasificador 3 (Verde): Edad y NPITOTAL.
- Clasificador 4 (Celeste): Años de educación y MOCA.
- Clasificador 5 (Rosa): Edad, años de educación y FAQ.

Tabla 22. Resultados del primer ensemble (SMV).

Clasificador	AUC	Error	CUI+	CUI-
Clasificador 1 (Azul)	0.8161	0.2296	0.4990	0.6251
Clasificador 2 (Rojo)	0.7534	0.2554	0.4517	0.6206
Clasificador 3 (Verde)	0.6381	0.4092	0.2497	0.4420
Clasificador 4 (Celeste)	0.7377	0.3428	0.2497	0.4420
Clasificador 5 (Rosa)	0.6754	0.3360	0.2497	0.4420
Ensemble	0.8326	0.1628	0.6082	0.7145

Como se puede apreciar en la Tabla 22, se han combinado cinco clasificadores con un error que varía en un amplio rango del 20-40%. El CUI es aceptable en general para valores positivos, y moderado en negativos.

El ensemble resultante obtiene un AUC de 0.8326, disminuyendo muy significativamente los errores de los sistemas que lo componen y mejorando el índice de utilidad clínica, tanto positivo como negativo.

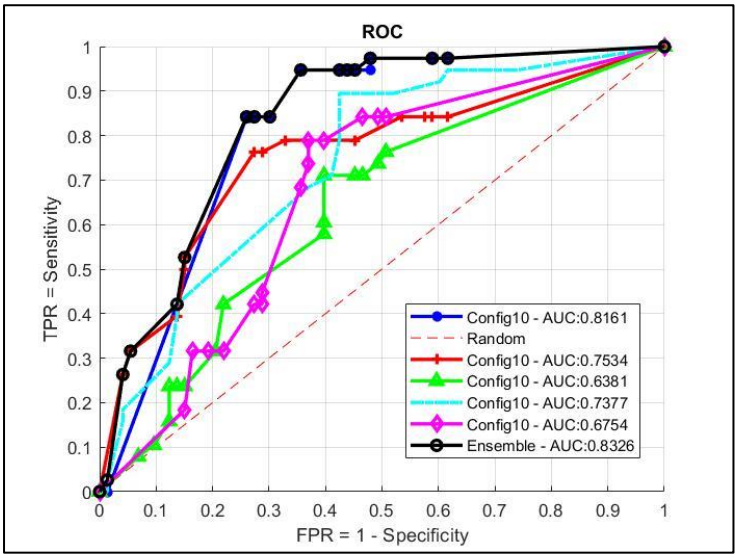

Figura 32. Curva ROC del ensemble y los clasificadores que lo componen (SMV).

Por la parte del graficado, igual que en el caso anterior, se modifica el AUC para la aproximación. Como se puede apreciar en la Figura 32, este se ajusta a los puntos máximos, que coinciden en su mayoría con el Clasificador 1 (azul).

Continuando con los mismos clasificadores, se ha modificado la estrategia de combinación a la WMV. En este caso, se han dado los pesos teniendo en cuenta el desempeño de los clasificadores individuales en la prueba anterior, obteniéndose los resultados mostrados en la Tabla 23:

Tabla 23. Resultados obtenidos del segundo ensemble (WMV).

Clasificador	Pesos	AUC	Error	CUI+	CUI-
Clasificador 1 (Azul)	0.3	0.8161	0.2296	0.4990	0.6251
Clasificador 2 (Rojo)	0.2	0.7534	0.2554	0.4517	0.6206
Clasificador 3 (Verde)	0.1	0.6381	0.4092	0.2497	0.4420
Clasificador 4 (Celeste)	0.2	0.7377	0.3428	0.2497	0.4420
Clasificador 5 (Rosa)	0.2	0.6754	0.3360	0.2497	0.4420
Ensemble	-	0.8326	0.1970	0.5558	0.6462

Nuevamente, el peso no afecta al entrenamiento de los clasificadores individuales.

En cuanto al error, en el caso del voto mayoritario el ensemble aumenta muy ligeramente el error con respecto a la estrategia de combinación anterior puesto que ahora hay clasificadores con mayor influencia que otros y estos se encuentran más equilibrados. Su AUC final, sin embargo, se mantiene.

El CUI, además, ha empeorado para la detección de casos positivos y negativos, aunque se mantienen ambos en un índice moderado.

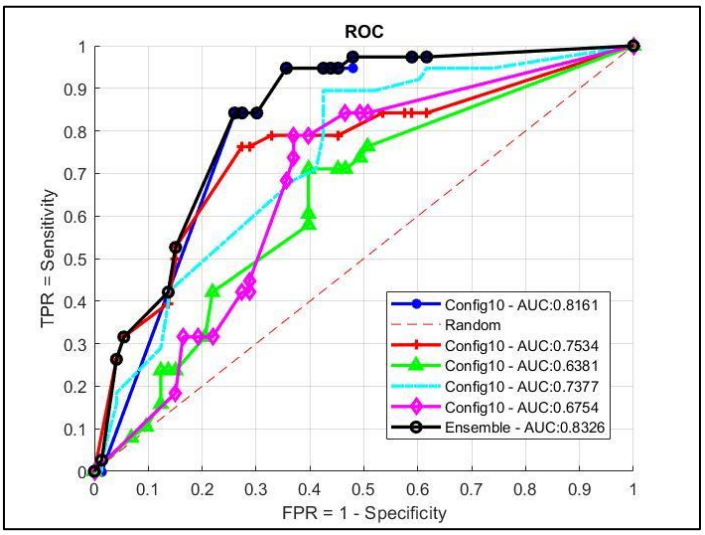

Figura 33. Curva ROC del ensemble y los clasificadores que lo componen (WMV).

En la Figura 33 se muestra la curva ROC. Igual que en el caso anterior, se modifica el AUC para la aproximación con el mismo método, por lo que se mantiene el resultado.

Capítulo 4: Conclusiones y trabajos futuros

En este capítulo se exponen las conclusiones obtenidas tras discutir los resultados, además de identificar ciertas líneas de investigación futuras.

Conclusiones

Siguiendo el objetivo planteado de estudiar las redes neuronales híbridas y los ensembles neuronales para asistir a la detección de DCL, se han desarrollado dos sistemas de detección con la ayuda de la herramienta *Kohonen and CPANN Toolbox.* Analizando los resultados de dichas implementaciones, se han obtenido las siguientes conclusiones:

- Se ha desarrollado un sistema basado en la red neuronal artificial híbrida *Counterpropagation*, con el fin de detectar el DCL frente a un paciente sin demencia. Se consiguen varios clasificadores óptimos con un buen índice de utilidad clínico, lo que indica que son apropiados para un diagnóstico real.
- Se ha encontrado una combinación óptima de criterios clínicos, que permiten realizar una detección adecuada. Además, gracias al uso de dos conjuntos de datos con criterios distintos, se ha comprobado que los tests neuropsicológicos MMSE y GDS tienen una mayor efectividad que MOCA y NPI para el problema abordado.
- Al experimentar con los parámetros de la red *Counterpropagation*, se ha encontrado que sus resultados varían bastante en comparación, pero se trata de una red bastante estable de manera general. Esto implica que es una buena opción para la implementación del sistema de detección.
- Se ha desarrollado un sistema de ensembles neuronales basados en la red *Counterpropagation*, que ha permitido la comparación de varios métodos de combinación, votación por mayoría simple (SMV, del inglés *Simple Majority Voting)* y votación por mayoría ponderada (WMV, del inglés *Weighted Majority Voting*), en busca de mejorar el rendimiento de un clasificador individual con el conjunto de varios con rendimiento mediocre. Con esta nueva estructura, se ha logrado reducir el error de clasificación con respecto al clasificador individual, lo que aporta una detección más fiable en caso de tener combinaciones de criterios clínicos no óptimas.
- La métrica del cálculo del AUC es útil para determinar la probabilidad que tiene un clasificador de determinar un caso positivo de DCL. La implementación escogida para la elaboración de la curva ROC, de la cual depende intrínsecamente el AUC, en un ensemble neuronal no aporta una visión clara a la hora de comparar sistemas que utilicen distintas estrategias de combinación puesto que dependen únicamente de los clasificadores individuales.

Trabajos futuros

El pilar central de este TFT es la creación de un sistema de detección de la DCL para un problema de clasificación binario. Lo cual despierta interés en otros puntos que podrían trabajarse en un futuro, descritos a continuación:

- Elaborar un sistema de ensembles más flexible que permita estudiar otros métodos de generación de diversidad y, también, explorar otras reglas de combinación de carácter empírico para el ensemble.

- La implementación de un sistema que pueda tratar otros problemas binarios como, por ejemplo, fase temprana de DCL (EMCI, del inglés *Early Mild Cognitive Impairment*) con fase tardía de DCL (LMCI, del inglés *Late Mild Cognitive Impairment*). O incluso problemas ternarios, que suelen añadir complejidad para la detección, tales como pacientes cognitivamente sanos (CN), DCL y EA, o CN, EMCI y LMCI.
- Hacer un tratamiento previo de los datos más exhaustivo por medio del uso de estadísticos para incluir una mayor cantidad o criterios clínicos más adecuados, y enriquecer al clasificador.
- El desarrollo de un sistema que utilice datos longitudinales para el estudio de la evolución de la demencia hacia la EA.
- Mejoras en la creación de la curva ROC para los ensembles neuronales, implementando un método suavizado que refleje unos resultados más cercanos al desempeño global según la estrategia de combinación elegida. Esto representaría una continuación directa de este estudio.

Bibliografía

A. K. Jain, Jianchang Mao and K. M. Mohiuddin. 1996. “Artificial Neural Networks: A Tutorial.” *Computer* 29 (3): 31–44. doi:10.1109/2.485891.

Alzheimer’s Association. 2021. “Causas y Factores de Riesgo.” *Alzheimer’s Association*. https://www.alz.org/alzheimer-demencia/causas-y-factores-de-riesgo?lang=es-MX.

Báez, Patricio García, Carmen Paz Suárez Araujo, Carlos Fernández Viadero, and Aleš Procházka. 2012. “Differential Diagnosis of Dementia Using HUMANN-S Based Ensembles.” In *Recent Advances in Intelligent Engineering Systems*, edited by János Fodor, Ryszard Klempous, and Carmen Paz Suárez Araujo, 378:305–324. Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-23229-9_14.

Ballabio, Davide, Viviana Consonni, and Roberto Todeschini. 2009. “The Kohonen and CP-ANN Toolbox: A Collection of MATLAB Modules for Self Organizing Maps and Counterpropagation Artificial Neural Networks.” *Chemometrics and Intelligent Laboratory Systems* 98 (2): 115–122. doi:10.1016/j.chemolab.2009.05.007.

Ballabio, Davide, and Mahdi Vasighi. 2012. “A MATLAB Toolbox for Self Organizing Maps and Supervised Neural Network Learning Strategies.” *Chemometrics and Intelligent Laboratory Systems* 118 (August): 24–32. doi:10.1016/j.chemolab.2012.07.005.

Cabrera-Leon, Ylermi, Patricio Garcia Baez, Juan Ruiz-Alzola, and Carmen Paz Suarez-Araujo. 2018. “Classification of Mild Cognitive Impairment Stages Using Machine Learning Methods.” In *2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES)*, 000067–000072. Las Palmas de Gran Canaria: IEEE. doi:10.1109/INES.2018.8523858.

Carlos Bonilla. 2019. “Inteligencia Artificial Para La Salud.” Blog. *El Hospital*. https://www.elhospital.com/blogs/Inteligencia-artificial-para-el-sector-de-la-salud+130190#:~:text=Inteligencia%20artificial%20para%20el%20sector%20de%20la%20salud&text=Hoy%20en%20día%2C%20la%20IA,el%20agotamiento%20de%20los%20médicos.

Cottrell, Marie, Madalina Olteanu, Fabrice Rossi, and Nathalie Villa-Vialaneix. 2018. “Self-Organizing Maps, Theory and Applications,” 21.

DementiaBank consortium group. 2017. “Dementia TalkBank.” Base de datos. https://dementia.talkbank.org/.

Dementias Platform UK. 2020. “DPUK Data Portal.” Accessed December 15. https://portal.dementiasplatform.uk/.

DRYAD. 2009. “Dryad Digital Repository.” Repositorio. https://dash.ucr.edu/search?q=alzheimer.

Edmond Teng 1, Brian W Becker, Ellen Woo, David S Knopman, Jeffrey L Cummings, Po H Lu. 2010. “Utility of the Functional Activities Questionnaire for Distinguishing Mild Cognitive Impairment from Very Mild Alzheimer Disease,” December, 348–353. doi:10.1097/WAD.0b013e3181e2fc84.

Equipo docente Grupo Sinapsis. 2019. “7 Ventajas Del Diagnóstico Temprano En La Enfermedad de Alzheimer.” https://www.gsinapsis.com/7-ventajas-del-diagnostico-temprano-en-la-enfermedad-de-alzheimer/.

Fawcett, Tom. 2006. “An Introduction to ROC Analysis.” *Pattern Recognition Letters* 27 (8): 861–874. doi:10.1016/j.patrec.2005.10.010.

Freeman, James A., and David M. Skapura. 1991. *Neural Networks: Algorithms, Applications, and Programming Techniques*. Computation and Neural Systems Series. Reading, Mass: Addison-Wesley.

Gobierno de Canarias. 2019. “Protocolo de Diagnóstico Del Deterioro Cognitivo En La Comunidad Autónoma de Canarias.”

https://www3.gobiernodecanarias.org/sanidad/scs/content/b3186014-4274-11ea-bbdf-d73a8968efc2/ProtocoloDC_Canarias.pdf.
Google Developers. 2020. “Google Developers, Machine Learning: Crash Course.” https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc.
Grossberg, Stephen. 2020. “Developmental Designs and Adult Functions of Cortical Maps in Multiple Modalities: Perception, Attention, Navigation, Numbers, Streaming, Speech, and Cognition.” https://www.frontiersin.org/articles/10.3389/fninf.2020.00004/full#B79.
Guo, Shuhan, Yanru Liu, Pengyun Wang, Sen Zhang, Buxin Han, Juan Li, and Wendong Xiao. 2021. “Recognition of Mild Cognitive Impairment in the Elderly Based on Machine Learning.” In *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*, 433–436. Beijing, China: IEEE. doi:10.1109/CISCE52179.2021.9445903.
Hajare Akash. 2021. “Hold Out Method & Random Sub-Sampling Method.” Blog. *Inblog*. https://inblog.in/Hold-Out-Method-Random-Sub-Sampling-Method-3MLDEXAZML.
J L Cummings 1, M Mega, K Gray, S Rosenberg-Thompson, D A Carusi, J Gornbein. 1994. “The Neuropsychiatric Inventory: Comprehensive Assessment of Psychopathology in Dementia,” December. doi:10.1212/wnl.44.12.2308.
Jason Brownlee. 2020a. “Why Data Preparation Is so Important in Machine Learning.” Blog. *Machine Learning Mastery*. https://machinelearningmastery.com/data-preparation-is-important/.
Jason Brownlee. 2020b. “What Is the Difference between Test and Validation Datasets?” Blog. *Machine Learning Mastery*. https://machinelearningmastery.com/difference-test-validation-datasets/.
José Francisco López. 2017. “Desviación Estándar o Típica.” Blog. *Economipedia*. https://economipedia.com/definiciones/desviacion-tipica.html#:~:text=La%20desviación%20estándar%20o%20desviación,mayor%20o%20igual%20que%20cero.
Joseph Rocca. 2019. “Ensemble Methods: Bagging, Boosting and Stacking.” https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205.
Julianna Delua. 2021. “Supervised vs Unsupervised Learning: What’s the Difference?” *IBM*. https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning.
Know Alzheimer - Neurología. 2014. “¿Cómo de Importante Es La Detección Temprana Del Alzheimer?” Blog. *Know Alzheimer*. https://knowalzheimer.com/como-de-importante-es-la-deteccion-temprana-del-alzheimer-que-puedo-hacer-si-sospecho-que-un-familiar-cercano-puede-padecerla/.
Kohonen, Teuvo. 1989. *Self-Organization and Associative Memory*. Vol. 8. Springer Series in Information Sciences. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-88163-3.
Liu, Xiaonan, Kewei Chen, Teresa Wu, David Weidman, Fleming Lure, and Jing Li. 2018. “Use of Multimodality Imaging and Artificial Intelligence for Diagnosis and Prognosis of Early Stages of Alzheimer’s Disease.” *Translational Research* 194 (April): 56–67. doi:10.1016/j.trsl.2018.01.001.
M F Folstein, S E Folstein, P R McHugh. 1975. “‘Mini-Mental State’. A Practical Method for Grading the Cognitive State of Patients for the Clinician,” November, 189–198. doi:10.1016/0022-3956(75)90026-6.
María Estela Raffino. 2020. “Neurona.” Diccionario. *Concepto.De*. https://concepto.de/neurona/.
MatLab. 2006. “Documentación MatLab: Función Trapz.” https://es.mathworks.com/help/matlab/ref/trapz.html#bua4lsr.

MatLab. 2020. “Cell Array.” Documentación. *MathWorks*. https://es.mathworks.com/help/matlab/cell-arrays.html.

Mato-Abad, Virginia, Isabel Jiménez, Rafael García-Vázquez, José Aldrey, Daniel Rivero, Purificación Cacabelos, Javier Andrade-Garda, Juan Pías-Peleteiro, and Santiago Rodríguez-Yáñez. 2018. “Using Artificial Neural Networks for Identifying Patients with Mild Cognitive Impairment Associated with Depression Using Neuropsychological Test Features.” *Applied Sciences* 8 (9): 1629. doi:10.3390/app8091629.

Michael W. Weiner. 2017. “Alzheimer’s Disease Neuroimaging Initiative (ADNI).” http://adni.loni.usc.edu/about/.

Milin, Petar, Harish Tayyar Madabushi, Michael Croucher, and Dagmar Divjak. 2020. “Keeping It Simple: Implementation and Performance of the Proto-Principle of Adaptation and Learning in the Language Sciences.” University of Birmingham. doi:10.25500/EDATA.BHAM.00000449.

Mitchell, Alex J. 2008. “The Clinical Significance of Subjective Memory Complaints in the Diagnosis of Mild Cognitive Impairment and Dementia: A Meta-Analysis.” *International Journal of Geriatric Psychiatry* 23 (11): 1191–1202. doi:10.1002/gps.2053.

Mónica Aranda and Alejando Calabria. 2019. “Impacto Económico-Social de La Enfermedad de Alzheimer.” Blog. https://blogcrea.imserso.es/impacto-economico-social-de-la-enfermedad-de-alzheimer/.

Mukhtar, G., and S. Farhan. 2020. “Convolutional Neural Network Based Prediction of Conversion from Mild Cognitive Impairment to Alzheimer’s Disease: A Technique Using Hippocampus Extracted from MRI.” *Advances in Electrical and Computer Engineering* 20 (2): 113–122. doi:10.4316/AECE.2020.02013.

National Center For Biotechnology Information. 2016. “NCBI Dataset Browser.” Base de datos. https://www.ncbi.nlm.nih.gov/.

NHS Digital. 2018. “NHS Digital Collected Databases.” NHS Digital. https://digital.nhs.uk/data-and-information/publications/statistical/recorded-dementia-diagnoses/september-2018.

NIA. 2016. “NIAGADS.” Base de datos. https://www.niagads.org/data.

Pacific Biosciences of California, Inc. 2016. “PacBIO.” Repositorio. https://downloads.pacbcloud.com/public/dataset/Alzheimer_IsoSeq_2016/.

Python Software Foundation. 2021. “Python 3.7.7.” *Python*. https://www.python.org/downloads/release/python-377/.

R. Hecht-Nielsen. 1987. “Counterpropagation Networks.” *Applied Optics* 16: 4979–4984. doi:10.1364/AO.26.004979.

Real Academia Española. 2005. “Inestabilidad.” *Diccionario Panhispánico de Dudas*. https://www.rae.es/dpd/inestabilidad.

Real Academia Española. 2021a. “Demencia.” *Diccionario de La Lengua Española*. Accessed February 21. https://dle.rae.es/demencia.

Real Academia Española. 2021b. “Anamnesis.” *Diccionario de La Lengua Española*. Accessed June 16. https://dle.rae.es/anamnesis.

Robi Polikar. 2009. “Ensemble learning.” *Scholarpedia*. http://www.scholarpedia.org/article/Ensemble_learning#Ensemble_combination_rules.

Sara López Álava. 2021. “Un Diagnóstico Precoz de La Demencia Mejora La Calidad de Vida Del Paciente.” *Rioja Salud*. Accessed July 12. https://www.riojasalud.es/saludable/protagonistas/un-diagnostico-precoz-de-la-demencia-mejora-la-calidad-de-vida-del-paciente.

Spyder IDE. 2020. “Spyder 4.1.3.” *Spyder*. https://www.spyder-ide.org/.

Stamate, Daniel, Richard Smith, Ruslan Tsygancov, Rostislav Vorobev, John Langham, Daniel Stahl, and David Reeves. 2020. “Applying Deep Learning to Predicting Dementia and Mild Cognitive Impairment.” In *Artificial Intelligence Applications and Innovations*,

edited by Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis, 584:308–319. IFIP Advances in Information and Communication Technology. Cham: Springer International Publishing. doi:10.1007/978-3-030-49186-4_26.

STIMULUS SALUD, S.A. 2017. “Inteligencia Artificial Para Detectar El Alzheimer.” Blog. https://stimuluspro.com/blog/inteligencia-artificial-para-detectar-el-alzheimer/.

The MathWorks, Inc. 2020. “MatLab.” *MathWorks*. https://es.mathworks.com/products/matlab.html.

Vikashraj Luhaniwal. 2020. “A Comprenhensive Guide to Feature Selection Using Wrapper Methods in Python.” Blog. *Analytics Vidhya*. https://www.analyticsvidhya.com/blog/2020/10/a-comprehensive-guide-to-feature-selection-using-wrapper-methods-in-python/.

Vilariño, Fernando, Ludmila I. Kuncheva, and Petia Radeva. 2006. “ROC Curves and Video Analysis Optimization in Intestinal Capsule Endoscopy.” *Pattern Recognition Letters* 27 (8): 875–881. doi:10.1016/j.patrec.2005.10.011.

Wang, Naibo, Jinghua Chen, Hui Xiao, Lei Wu, Han Jiang, and Yueping Zhou. 2019. “Application of Artificial Neural Network Model in Diagnosis of Alzheimer’s Disease.” *BMC Neurology* 19 (1): 154. doi:10.1186/s12883-019-1377-4.

Xabier Basogain Olabe. 2008. *Redes Neuronales Artificiales y Sus Aplicaciones*. https://ocw.ehu.eus/pluginfile.php/40137/mod_resource/content/1/redes_neuro/contenidos/pdf/libro-del-curso.pdf.

XNAT. 2007. “OASIS Brains Datasets.” Base de datos. https://www.oasis-brains.org/#about.

Yoav Freund and Robert E. Schapire. 1996. “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting” 904 (December). https://www.face-rec.org/algorithms/Boosting-Ensemble/decision-theoretic_generalization.pdf.

Ziad S Nasreddine 1, Natalie A Phillips, Valérie Bédirian, Simon Charbonneau, Victor Whitehead, Isabelle Collin, Jeffrey L Cummings, Howard Chertkow. 2005. “The Montreal Cognitive Assessment, MoCA: A Brief Screening Tool for Mild Cognitive Impairment,” April, 695–699. doi:10.1111/j.1532-5415.2005.53221.x.