



UNIVERSIDAD DE GRANADA

Facultad de Ciencias

Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones

DOBLE GRADO EN INGENIERÍA
INFORMÁTICA Y MATEMÁTICAS

TRABAJO FIN DE GRADO
Desarrollo de sistemas
inteligentes para la
clasificación automática de
la enfermedad del
Parkinson utilizando
imágenes MRI

Presentado por:
D. Luis Balderas Ruiz

Tutor:
Catedrático D. Ignacio Rojas Ruiz

*“How do we fight with this beast? Living with
an incurable disease like the Parkinson’s is very
different from living with a terminal illness.
This is a disease that you will live with for a long
time. You really have to be at peace with that”*
Connie Carpenter-Phinney

Agradecimientos

Quiero dedicarle mis primeras palabras a mi tutor, Ignacio Rojas Ruiz, por todo el apoyo, ánimo y confianza que ha depositado en mí para llevar a cabo este ambicioso proyecto. Ha sido un verdadero honor. De igual manera, a Alberto García Salguero por haberme ayudado tanto y no cansarse de contestar mis preguntas. Parte de esto no habría sido posible sin ti.

A todos mis profesores y especialmente a Daniel Sánchez, Jesús Chamorro, Fernando Berzal, Juan Huete, José Manuel Benítez, Alberto Guillén, Juan Julián Merelo y Javier Merí por tantos y tantos consejos, tanta dedicación y tantas enseñanzas a lo largo de estos cinco años. Si algo me llevo a parte de algún conocimiento y mis compañeros es modelos de conducta en los que fijarme. Ojalá algún día sea un profesional de tanta valía como lo sois vosotros.

Gracias a mis padres, artífices de todo, incombustibles ante los problemas y las frustraciones que he ido experimentando y, sin duda alguna, mentores en una nueva vida que se abre camino en breve. Espero ser algún día padre y transmitir todo lo que vosotros me habéis dado.

A mis amigos, que me distéis aliento para no bajar los brazos cuando ya no quedaban apenas fuerzas. Nacho, Natalio, Luis, Alberto, Pablo, Diego, Darío, Rubén, Gema y un montón más que me dejó en el tintero. Sois verdaderos catalizadores para mí.

Y a ti, Maripaz. En especial a ti. Por hacerme sentir capaz en cada momento, por animarme a dar todo lo que llevo dentro y a seguir, seguir siempre. Tu tesón y tu capacidad no tienen límite. Eres un ejemplo para mí y no pasa un día sin que dé gracias a la vida por habernos regalado aquel momento en el que nos encontramos. Te quiero muchísimo.

Granada, 10 de junio de 2019

Design of intelligent systems for Parkinson's Disease classification using MRI imaging

ABSTRACT

The aim of this project is to study the potential for MRI evaluation of Parkinson's Disease, performing experiments to test the effectiveness of different algorithms (SVM, stacking, Deep Learning) and methods in the disease's identification. More specifically, the objective of this project is to develop an automated software that could help to classify patients of Parkinson's disease and aid in the discovery of new regions affected by PD or, for that matter, any neurological disease.

Nowadays, brain disease and dementia research is one of the most important fields of investigation. According to the Parkinson's Foundation, 46.8 million people worldwide are living with some form of dementia. Dementia is the condition characterized by the deterioration of brain functions. This loss of faculties leads to impairment in daily life and can even lead to loss of consciousness. Particularly, it is estimated that there are more than 10 million people worldwide now living with Parkinson's Disease (PD). In this project we explain carefully the different stages of PD. Over the past two decades, various objective measures have been adopted for the differential diagnosis of PD, including electrophysiological and neurophysiological tests. However, the most developed area in providing an objective assessment is neuroimaging. This method involves Magnetic Resonance Imaging (MRI) and computational algorithms. We focus our project in this kind of techniques to build a robust Computer-Aided Diagnosis system.

We divide the brain 3D images in slices (2D images) to study the different brain's regions. To discover which slices of the MRI are the most relevant in the brain for identification of Parkinson's Disease region, we intend to test different feature extraction and feature selection methods. Most of the research is oriented to slices and two-dimensionality transforms. When working with MRI images, which have a number of slices of 157x189x136 in planes X,Y,Z respectively, we have 4035528 voxels with 255 possible values. We would thus find more than one billion different feature vectors. We need to reduce the amount of resources required to describe such a large dataset. Usually, a very efficient strategy to extract features from images is to convert the images to a different domain. Fourier's Transform would bring us a representation of the frequencies making up the image, thus highlighting some information which could be otherwise hidden and inaccessible. The main problem stemming from the utilization of Fourier's Transform is that some information is lost, specifically the time-domain information. It would be a better approach to use a transformation that didn't lose information. The most common in MRI studies is Wavelet transform (2D), which does not have the same problem of loss of time information as Fourier. Therefore, we select discrete wavelet transform as our tool to extract features from MRI images. After feature extraction, we have a matrix of vectors of features which represents our initial data set. These vectors have thousands of features, each describing a specific characteristic of the elements which they represent. As a result of this huge amount of information, redundancy might be a tough issue. In order to address it, we choose Principal Component Analysis as a powerful statistical tool to reduce the dimensionality and remove redundancy at the same time that the most relevant information is kept. In particular, we set 95% of the amount of variance as the minimum percentage that the number of components given by PCA needs to explain. All these techniques conform the data preprocessing phase. Prior to any manipulation or information extraction, we need to normalize the whole data set to represent the same reality. Every MRI image should represent the same space, which means that each region of each image has to describe the same brain region. The ventricles should be placed in the same coordinates for each image of each patient. Given that brains vary very much from subject to subject, this is a hard goal to achieve because different genetic inheritance and life experiences to each brain make having its own peculiarities. So, we have to take into consideration two main objectives to be accomplished. The first one is normalize the gray color levels of each image so that

all of them belong to the same scale. The second one is reshape the image to the same bounding box so that all brains have the same measures. To realize the normalization we used Statistical Parametric Mapping (SMP 12), which is an algorithm that analyses each voxel using a standard statistical test and then reassembles parameters into a spatially normalized image. Taking into account that we have two kind of subjects, patients who suffer Parkinson's Disease and healthy patients, the explained procedures were combined in different ways to reach the hypothesized results and to perform four main experiments:

- The first experiment was to determine which plane of the MRI image is more important to classify subjects. In MRI images, there are three planes of acquisition: X (Axial), Y (Coronal) and Z (Sagittal). To reduce computational time and wasted slices, we selected a plane to extract slices. We have found that Coronal plane is the best one to study when it comes to diagnose Parkinson's Disease.
- After we decided which plane of X,Y,Z (Y, as we have just said) we were going to use, we wanted to know the accuracy of each slice in order to select which slices of the brain could be most suitable to extract features from and to base a classification on. This will be our first approach to find the best slices in diagnose.
- We compared the use of white matter or gray matter instead of the whole matter to improve the performance of the classification. However, the results were not conclusive, so we decided to carry on using the whole matter as our source of information in order to find the best slices.
- A first optimization of relevant slices, Coronal slices (Y plane), based on Ensemble Learner methods, in particular Stacking, was carried out. First, after dividing the data set in training set and test set, we developed a first phase in which 189 SVM models are trained (one per each Coronal slice). 10-fold cross validation was used to avoid overfitting. Besides, a grid search was made to tune SVM hyperparameters (C and γ). As a result, by the end of this phase, we had a matrix in which each row was a patient (from the test set) and each column refers to a slice, i.e., we had a $\text{num_patients} \times 189$ matrix. Each element of the matrix is a prediction corresponding to the patient in the row as a result of the study in each slice. For example, we could have a row like this:

Patient 1 → (0,0,0,1,0,...,0), where 0 means the patient is ill and 1, healthy.

We know the real state of the patient (the system doesn't know it at all). Taking this into account, we carried out the second phase, where we try to find which slices are the most important when it comes to diagnose the real state (which is known for us) of the patient. To assure the interpretability of the results, we used a linear model (Logistic Regression) to study which columns (slices) are most relevant in the linear combination to generate the label. We chose the absolute value of the coefficients as an indicator of the importance. This procedure and the results are deeply explained in Section 4.3. We studied the 10 best slices to take into consideration for diagnosing Parkinson's Disease. The precision values obtained are greater than 72%. In addition, the recall values obtained are also greater than 80%.

- In the second optimization, which had the same structure, we established a new way of facing neurodegenerative illnesses' diagnose. We use Deep Learning, in particular Convolutional Neural Network, as the model of the first phase, i.e., the model to learn the physical features of Parkinson's Disease in a Magnetic Resonance Image. In this case, no data preprocessing was carried out in order to take advantage of the power of deep learning. We designed the topology of the net using convolutional layers, dropout layers and pooling

layers. We fixed the number of epochs to 100 and we used early stopping to reduce computational time when the neural network doesn't seem to be learning new features. After different simulations, the final topology of the net is: A 2D convolutional layer (320 parameters), a pooling layer, a 2D convolutional layer (4624 parameters), a pooling layer, a 2D convolutional layer (2320 parameters), a pooling layer and dropout layers (dense to finish). As in the first optimization, we studied the 10 best slices (images were specified in Section 4.3 and 4.4 to visualize the slices) to find out Parkinson's Disease. However, results were not as promising as in the first approach, so we have found that deep learning is not suitable for problems with too little amount of data.

Furthermore, we were very interested in developing each concept and tool used mathematically, in order to guarantee the rigor needed in this challenging issue. The precision values obtained are greater than 72%. In addition, the recall values obtained are also greater than 80%. This suggests that the proposed workflow and its application could help in the investigation of Parkinson's disease and aid in the research of other neurodegenerative diseases, improving their diagnostic accuracy and identifying the most relevant zones and regions of the brain associated with each disease.

KEY WORDS

Parkinson's Disease (PD), Feature Extraction, Discrete Wavelet Transform (DWT), Feature Selection, Principal Component Analysis (PCA), Classification, Support Vector Machine (SVM), Logistic Regression, Stacking, Ensemble Learner, Deep Learning, Convolutional Neural Networks (CNN), Pooling, Dropout, Convolution.

Diseño de sistemas inteligentes para la clasificación automática de la enfermedad del Parkinson utilizando imágenes MRI

RESUMEN

El objetivo de este proyecto es estudiar la potencialidad de las imágenes de resonancia magnética (MRI) en la evaluación de la enfermedad del Parkinson, llevando a cabo experimentos y probando la efectividad de distintos algoritmos (SVM, stacking, Deep Learning) y métodos para el diagnóstico de la enfermedad.

Para la extracción de características de las imágenes utilizamos la Transformada Discreta Wavelet 2D, seguido de Análisis de Componentes Principales (PCA) para la selección y reducción de características. Después, aplicamos Máquinas de Soporte Vectorial (SVM), Regresión logística y Redes Neuronales Convolucionales para la clasificación y optimización.

Para determinar qué cortes de la imagen MRI son los más relevantes en el cerebro para identificación del Parkinson, una optimización basada en Ensemble Learners, más concretamente Stacking, ha sido llevada a cabo. Además, introducimos las Redes Neuronales Convolucionales como una nueva alternativa en la batalla que supone el diagnóstico precoz de este tipo de demencia. Llevamos a cabo una clasificación entre pacientes enfermos de Parkinson y pacientes sanos (grupo de Control), a fin de valorar nuestros resultados.

Por otra parte, nuestro objetivo ha sido desarrollar cuidadosamente el trasfondo matemático subyacente a las herramientas y conceptos usados con el objetivo de garantizar el rigor tan necesario ante un problema de tal calibre. Obtenemos unos valores de *precision* mayores al 72%. De la misma manera, obtenemos un *recall* mayor del 80%. Dichos resultados sugieren que la metodología de trabajo aplicada en este Trabajo Final de Grado pudieran ayudar en la investigación y diagnóstico del Parkinson, así como otras enfermedades neurodegenerativas identificando las zonas más relevantes del cerebro asociadas a la enfermedad.

PALABRAS CLAVE

Enfermedad del Parkinson (PD), Extracción de características, Transformada Discreta Wavelet (DWT), Selección de Características, Análisis de Componentes Principales (PCA), Clasificación, Máquinas de Soporte Vectorial (SVM), Regresión Logística, Stacking, Ensemble Learner, Deep Learning, Redes Neuronales Convolucionales (CNN), Pooling, Dropout, Convolución.

Índice

1 Contexto: Parkinson y Definición del Problema	6
1.1 Objetivos del proyecto	7
1.2 Experimentos	7
2 Enfermedad de Parkinson y sus estados	8
2.1 Contexto global	8
2.2 Corteza cerebral y ganglio basal	10
2.3 Estadios del Parkinson	12
2.4 Consecuencias en el cerebro	12
3 Conceptos y herramientas matemáticas	14
3.1 Resonancia magnética nuclear	14
3.2 Datos de entrada: PPMI	15
3.3 Normalización	16
3.4 Características	17
3.4.1 Extracción de características	17
3.4.2 Selección de características	34
3.5 Clasificación	38
3.5.1 SVM	38
3.5.2 Validación cruzada con K-particiones	42
3.5.3 Ajuste de hiperparámetros y GridSearch	43
3.5.4 Regresión logística	44
3.5.5 Optimización: En busca de los mejores cortes	46
3.5.6 Deep Learning y Stacking	46
4 Metodología y experimentos	52
4.1 Experimento 1: Determinando el mejor plano	53
4.1.1 Material y metodología	53
4.1.2 Resultados	53
4.2 Experimento 2	56
4.2.1 Material y metodología	56
4.2.2 Resultados	56
4.2.3 Conclusiones	59
4.3 Experimento 3	60
4.3.1 Material y metodología	60
4.3.2 Resultados	60
4.4 Experimento 4	66
4.4.1 Material y metodología	66
4.4.2 Resultados	68

5 Desarrollo del proyecto software	73
5.1 Análisis y diseño	73
5.1.1 Subsistemas	73
5.1.2 Requisitos de datos	73
5.1.3 Requisitos funcionales	73
5.1.4 Diagramas	74
5.2 Implementación	77
6 Conclusiones y proyectos futuros	78
6.1 Conclusiones	78
6.2 Proyectos futuros	78
7 Código	80
7.1 Normalización	80
7.2 Sujetos	81
7.3 Experimento 1	82
7.3.1 Resultados experimento 1	86
7.4 Experimento 2	88
7.4.1 Resultados experimento 2	91
7.5 Experimento 3	92
7.5.1 Resultados experimento 3	96
7.5.2 Clasificación con los mejores cortes. Experimento 3	97
7.6 Experimento 4	100
7.6.1 Resultados experimento 4	104
7.6.2 Clasificación con los mejores cortes. Experimento 4	105
8 Bibliografía	110

Índice de figuras

2.1. Susbtantia nigra	8
2.2. Mesencéfalo	8
2.3. Pérdida de la pigmentación	9
2.4. Cuerpos de Lewy en el citoplasma	9
2.5. Lóbulos de la corteza cerebral	10
2.6. Ganglio basal	11
2.7. MRI de un sujeto sano	13
2.8. MRI de un sujeto enfermo de PD. Véase la pérdida de materia gris	13
3.1. Funcionamiento de los campos magnéticos, spin y NMR	15
3.2. A la izquierda, imagen MRI sin normalizar. A la derecha, resultado de la normalización	16
3.3. Transformada ventana (izquierda) y transformada wavelet (derecha)	25
3.4. Wavelet Haar	28

3.5.	La transformada wavelet discreta 2D de una imagen a través de la wavelet Haar	32
3.6.	Esquema de actuación algorítmica para calcular transformada 2D	33
3.7.	Hiperplano separador con el margen y los vectores soporte (aquellos que definen la región)	39
3.8.	Hiperplano separador con el margen y clases solapadas	40
3.9.	Ejemplo de validación cruzada	43
3.10.	Modelo de stacking	46
3.11.	Comparación entre el cerebro humano y una red neuronal ([1])	47
3.12.	Ejemplo de red sencilla (izquierda) y red con capas ocultas (derecha) ([2])	48
3.13.	Ejemplo de pooling con máximo	50
3.14.	Ejemplo de convolución + pooling	50
3.15.	Ejemplo de dropout	51
4.1.	Resultados para el plano axial	54
4.2.	Resultados para el plano coronal	54
4.3.	Resultados para el plano sagittal	55
4.4.	Comparativa de accuracy entre C1 y C2	57
4.5.	Diferencia de accuracy entre C2 y C1 para cada slice	57
4.6.	Número de componentes principales por slice para C1	58
4.7.	Número de componentes principales por slice para C2	58
4.8.	Pesos asociados a cada slice en la regresión logística para el diagnóstico. Experimento 3	61
4.9.	Cortes más determinantes para Parkinson. Experimento 3	62
4.10.	Cortes más determinantes para Control. Experimento 3	63
4.11.	Matriz de confusión para la clasificación con los mejores cortes	64
4.12.	Curva ROC para la clasificación con los mejores cortes	64
4.13.	Configuración inicial de la red	66
4.14.	Diseño de la red, parte 1	67
4.15.	Diseño de la red, parte 2	68
4.16.	Pesos asociados a cada slice en la regresión logística para el diagnóstico. Experimento 4	69
4.17.	Cortes más determinantes para Parkinson. Experimento 4	70
4.18.	Cortes más determinantes para Control. Experimento 4	71
4.19.	Matriz de confusión para la clasificación con los mejores cortes	72
5.1.	Esquema de caja negra	74
5.2.	Esquema F-Armazón de los subsistemas	75
5.3.	Esquema de flujo de datos para Experimento 1	75
5.4.	Esquema de flujo de datos para Experimento 2	76
5.5.	Esquema de flujo de datos para Experimento 3	76
5.6.	Esquema de flujo de datos para Experimento 4	77

Índice de tablas

4.1. Resultados del experimento 1	55
---	----

1. Contexto: Parkinson y Definición del Problema

En la actualidad, las enfermedades neurodegenerativas son una de las afecciones más preocupantes para el ser humano y, como tal, es uno de los campos de investigación más importantes que existen. Según *Parkinson's Foundation* [3], 46.8 millones de personas en todo el mundo conviven con algún tipo de demencia. Estudios anteriores preveían que, en el año 2020, 42.3 millones de ciudadanos estarían afectados por estas enfermedades. Sin embargo, la ratio de enfermos se ha superado en más de 4 millones un año antes de la fecha esperada, lo que genera una preocupación acuciante. El mismo estudio pronosticó que el número de pacientes con demencia se duplicará en los próximos 20 años.

La demencia, a grandes rasgos, es un estado caracterizado por el deterioro de las funciones cerebrales. Este deterioro o pérdida de facultades da lugar a grandes inconvenientes en el día a día, llegando a extremos tan graves como la pérdida de la conciencia. Se estima que hay más de 10 millones de personas enfermos de Parkinson (en lo que sigue, PD) alrededor del mundo ([4]). Este hecho hace que la investigación de esta enfermedad en concreto sea muy relevante, dado que un diagnóstico precoz podría frenar el desarrollo de la misma. Desgraciadamente, actualmente no existe una cura para el Parkinson, pero sí hay medicamentos que inhiben su desarrollo, dándoles a los pacientes un mínimo de calidad de vida durante un periodo de tiempo más extenso.

Los principales métodos de diagnóstico se fundamentan en resultados clínicos, basados en la evaluación médica a través de distintas pruebas al paciente. El diagnóstico actual recae en la presencia de anomalías o disfunciones motoras, signo de que el paciente sufre indudablemente un PD en estado avanzado. En dicho estado, la terapia neuroprotectora apenas produce mejorías sustanciales en los pacientes, por lo que es verdaderamente importante encontrar biomarcadores objetivos y válidos que ayuden a distinguir entre pacientes enfermos de PD de la población sana.

En las dos décadas anteriores se adoptaron diversas medidas para el diagnóstico diferencial de PD, incluyendo pruebas olfativas, electrofisiológicas y neuropsicológicas [5]. Sin embargo, neuroimagen es el área más desarrollada para enfrentar diagnósticos. Estos métodos incluye la Imagen de Resonancia Magnética (MRI). MRI es una tecnología no invasiva con una gran resolución espacio-temporal y ha sido enormemente utilizado para el estudio de disfunciones cerebrales de todo tipo. La gran cantidad de información que MRI nos da sobre los tejidos ha mejorado de forma muy sustancial el diagnóstico de patologías cerebrales y su tratamiento. Es conveniente señalar que la basta cantidad de información que nos da está lejos de poder ser procesada manualmente, por lo que urge el desarrollo de herramientas de análisis automatizado. Dicha necesidad hace nacer este proyecto, basado en la extracción de características de imágenes cerebrales para la clasificación de sujetos en enfermos de PD o grupo de control con la mayor exactitud posible.

1.1. Objetivos del proyecto

El objetivo principal de este proyecto es diseñar y desarrollar un sistema avanzado que clasifique pacientes en enfermos y sanos tras analizar y refinar datos provenientes de MRI, así como descubrir qué zonas del cerebro son las más determinantes en el diagnóstico de la enfermedad.

Existen multitud de artículos en la literatura que llevan a cabo clasificación de pacientes enfermos y sanos. Dicha clasificación de enfermos de Alzheimer o Parkinson suelen estar basadas en la evaluación de las capacidades motoras de los individuos. Sin embargo, este enfoque sobre Parkinson utilizando MRI es novedoso. De la misma manera, utilizaremos métodos de extracción y selección de características, en particular transformada Wavelets 2D y PCA (Análisis de Componentes Principales). Este enfoque ha sido previamente sugerido por otros investigadores ([6], [7], [8], [9], [10], [11], [12], [13], [14], [15]).

Finalmente, la mayoría de las investigaciones exploran las regiones identificadas por expertos médicos. En este proyecto, nuestro interés es encontrar los planos más relevantes para la clasificación de enfermos de PD. Para ello, existen muchas técnicas de optimización disponible: Optimización por Colonia de Hormigas, Algoritmo de Búsqueda Gravitacional, algoritmo genético NSGA-II. En mi caso, enfoco el problema de una manera totalmente diferente utilizando un ensemble learner basado en Stacking, donde en las primeras capas utilizo SVM con GridSearch para configurar los hiperparámetros y en la segunda, regresión logística (en búsqueda de interpretabilidad).

1.2. Experimentos

En la realización del proyecto utilizamos diferentes algoritmos y métodos, incluyendo preprocesamiento de imágenes, extracción, selección de características, clasificación y optimización de los resultados. Los experimentos designados son los siguientes:

- El primer experimento fue determinar qué plano de una imagen MRI es más importante en la clasificación de los sujetos. Las imágenes MRI tienen tres planes: X (axial), Y (coronal) y Z (sagittal). Para reducir el tiempo computacional, seleccionamos el plano con los cortes más interesantes.
- En el segundo experimento, elegido ya el plano correspondiente, comparo el rendimiento de la materia gris, materia blanca y el materia completa para ahorrar costes y mejorar la clasificación.
- Para la extracción de características, utilzo la transformada discreta Wavelet en 2D.
- El tercer experimento consistirá en una optimización basada en stacking (SVM+Regresión logística) para determinar los mejores cortes del plano elegido.
- Por último, introducimos deep learning en la detección de la enfermedad del Parkinson para intentar encontrar los mejores cortes.

2. Enfermedad de Parkinson y sus estados

2.1. Contexto global

Parkinson's Foundation ([3]) describe el Parkison como sigue:

“ El Parkinson es un desorden neurodegenerativo que causa la muerte de las neuronas dopaminérgicas (neurotransmisores que producen y secretan dopamina) de un área concreta del cerebro llamada *substantia nigra pars compacta* (*SNpc*).”

Esta estructura se encuentra localizada en el mesencéfalo y debe su color y su nombre a la presencia de un pigmento llamado neuromelanina que se encuentra dentro de las neuronas que lo forman.

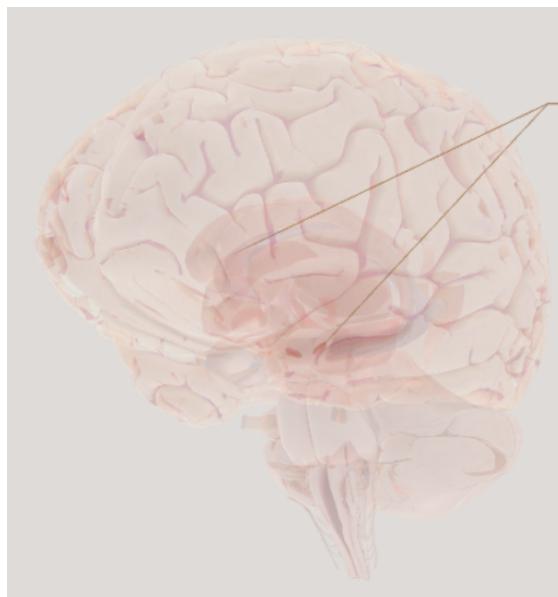


Figura 2.1: Susbtantia nigra



Figura 2.2: Mesencéfalo

Estas neuronas dopaminérgicas tienen principalmente la función de regular la actividad motora por medio de la síntesis y la secreción de dopamina, por lo que cuando mueren se manifiestan los típicos signos de la enfermedad que nos resultan familiares: temblores, lentitud en el movimiento (bradiquinesia), inestabilidad, caídas frecuentes... A nivel macroscópico esto se manifiesta en la pérdida de pigmentación característica de la SNpc.

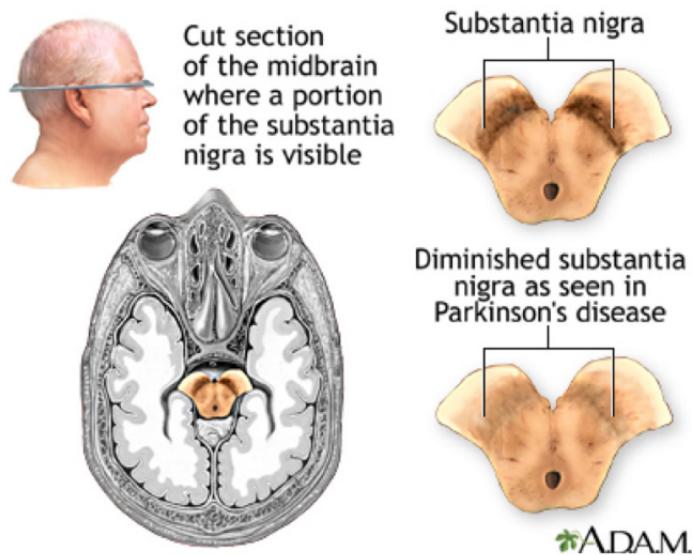


Figura 2.3: Pérdida de la pigmentación

Cabe destacar que en las neuronas supervivientes, a nivel microscópico se observan los característicos cuerpos de Lewy, que son unas “bolsitas” de proteínas que se acumulan en el citoplasma o cuerpo de la célula.

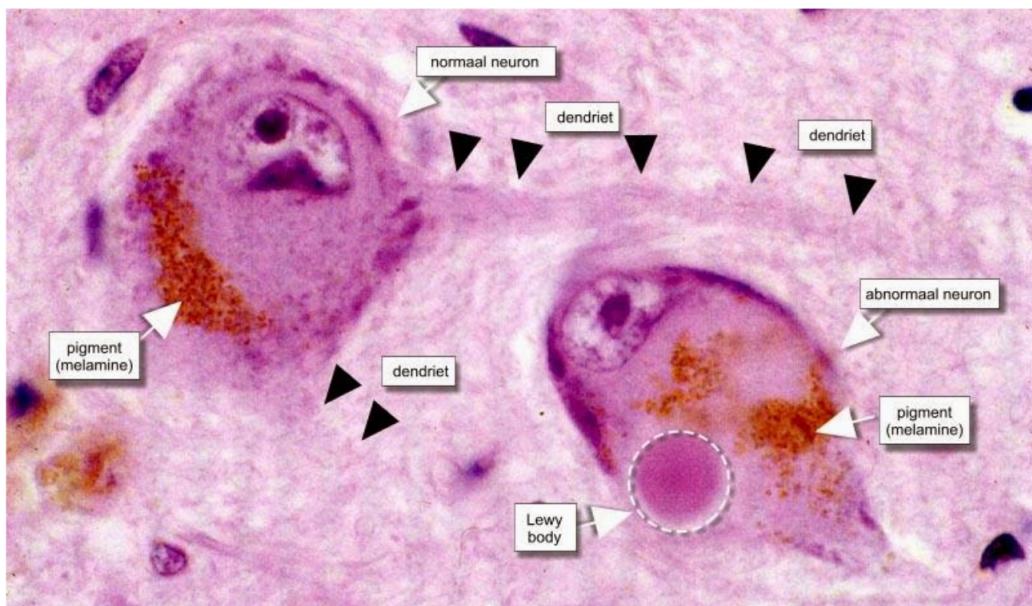


Figura 2.4: Cuerpos de Lewy en el citoplasma

2.2. Corteza cerebral y ganglio basal

La primera capa que nos encontramos al explorar el cerebro humano es la materia dura, esto es, una membrana que envuelve al cerebro, siendo la última capa de las meninges, cubriendo y por tanto protegiendo el cerebro y la médula espinal. Bajo esta membrana encontramos el cortex, formado por millones de neuronas de un color gris claro (la materia gris) organizadas en seis capas de entre dos y cuatro milímetros de grosor. La corteza cerebral juega un papel trascendental en la conciencia, el pensamiento, el lenguaje, la memoria, la percepción y la atención. La materia gris es una componente muy importante de nuestro sistema nervioso central. Por otra parte, la materia blanca está formada por axones que interconectan las neuronas en diferentes regiones de la corteza y del sistema nervioso central.

La corteza cerebral se divide en cuatro lóbulos:

- Lóbulo temporal: Clave en la percepción auditiva, comprensión del lenguaje, memoria y aprendizaje. Contiene el hipocampo.
- Lóbulo frontal: Corteza motora primaria, contiene también la mayoría de las neuronas dopaminérgicas en el cortex.
- Lóbulo parietal: esencial para la visión espacial, la navegación y el sentido del tacto.
- Lóbulo occipital: Cortex visual primario, responsable de la creación de los sueños.

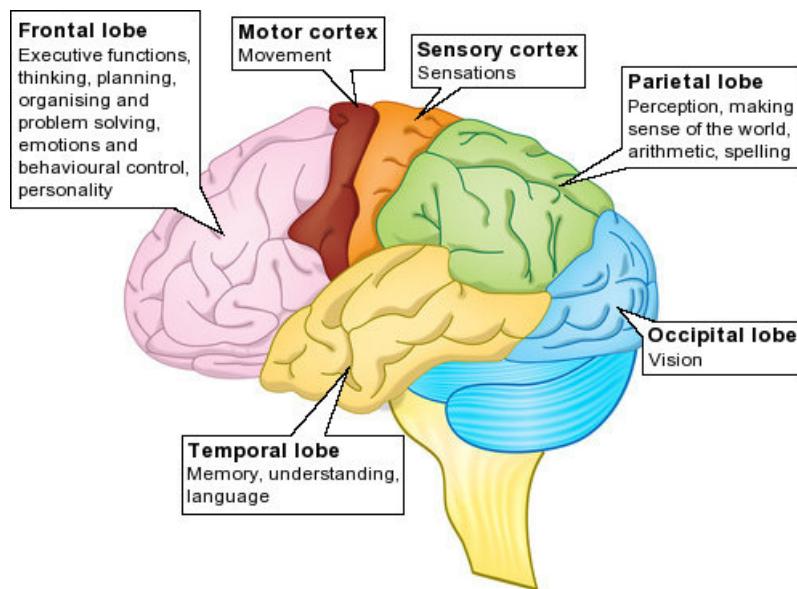


Figura 2.5: Lóbulos de la corteza cerebral

Las neuronas dopaminérgicas de la SNpc proyectan sus axones hacia el ganglio basal, formando así el sistema dopaminérgico nigroestriatal. El ganglio basal, que se encuentra en estrecha relación con la SNpc, está descrito como la estructura cerebral más afectada por PD. Cumple un papel esencial tanto en la ejecución de movimientos voluntarios como en actividades cognitivas, por lo que su deterioro asociado a PD afectará a estas funciones.

El ganglio basal puede verse afectado según el subtipo de la enfermedad. Hay algunos enfermos que sufren cambios microestructurales en la substantia nigra mientras que en otros apenas se aprecia. En general, el ganglio basal acaba por atrofiarse.

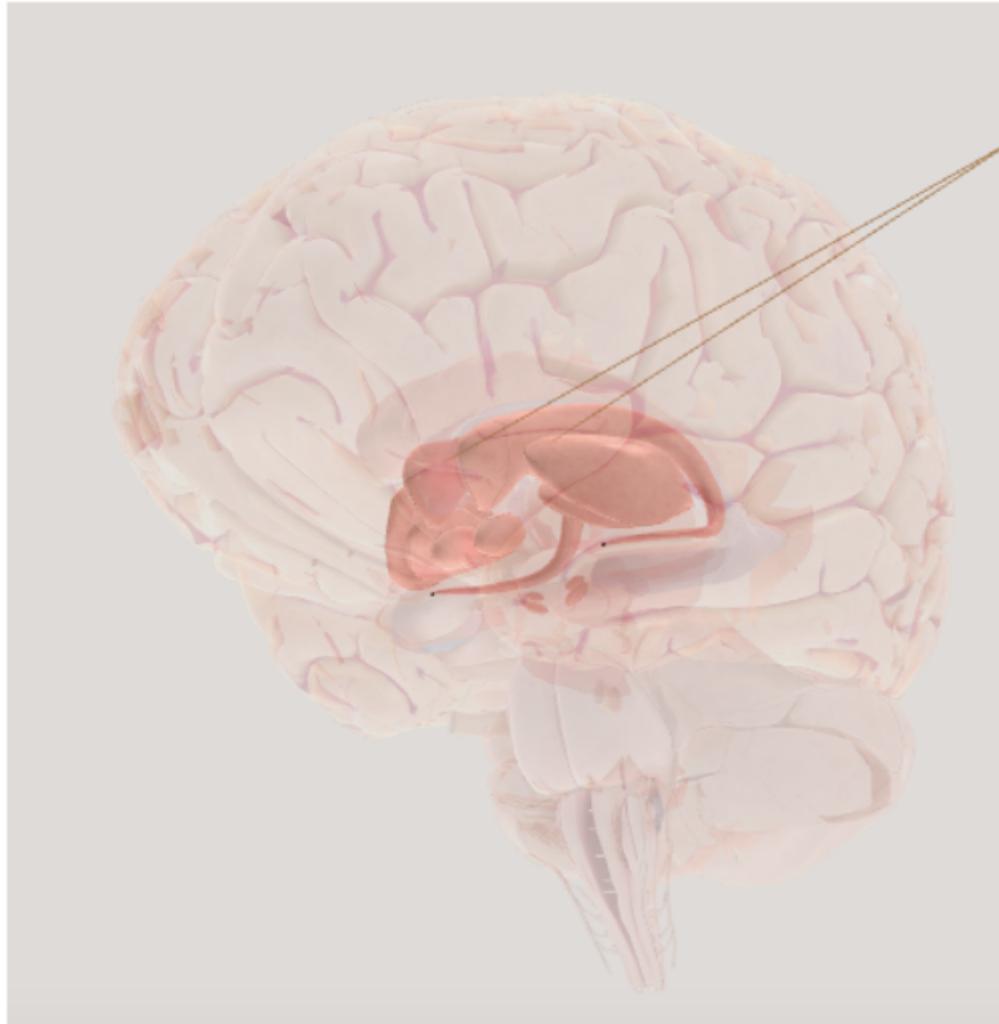


Figura 2.6: Ganglio basal

Según [4], los pacientes de PD muestran anisotropía fraccional reducida en la substantia nigra y aumento de la difusividad media y radial en la substantia nigra y el globo pálido

(parte del ganglio basal), cuyos efectos pueden verse en técnicas de imagen tales como tractografías.

2.3. Estadios del Parkinson

La enfermedad de Parkinson afecta al ser humano de muy distintas maneras. Los enfermos no tienen por qué sufrir los mismos síntomas y, si lo hicieran, tampoco tienen por qué experimentarlos en el mismo orden ni con la misma intensidad. Sin embargo, existen algunos patrones típicos en el progreso de la enfermedad divididos en estadios [4]:

- Estadio uno: Durante este estado inicial, la persona tiene síntomas menores que no interfieren en su vida diaria. Pueden darse temblores y movimientos involuntarios en un lado del cuerpo. De igual manera, se producen cambios posturales, en la forma de andar y en la expresión facial.
- Estadio dos: Los síntomas empeoran. Aparecen temblores, rigidez y movimientos involuntarios en ambos lados del cuerpo.
- Estadio tres: Considerado el estadio medio, se caracteriza por la ralentización de los movimientos y la pérdida del equilibrio. Las caídas empiezan a ser comunes.
- Estadio cuatro: En este punto, los síntomas son severos. Es posible permanecer de pie sin ayuda, pero en general se necesita un andador para desplazarse. La persona es incapaz de vivir sola y requiere asistencia.
- Estadio cinco: Este es el estadio más avanzado. Es imposible andar o ponerse de pie por la debilidad en las piernas. La persona requiere silla de ruedas y asistencia total para todas las actividades.

2.4. Consecuencias en el cerebro

Como hemos comentado, PD afecta a la substantia nigra. También reduce diferentes regiones de la materia gris en el lóbulo temporal. En la figura 2.7 podemos ver una resonancia magnética de un sujeto sano mientras que en la figura 2.8 vemos a un enfermo. Se puede observar la reducción de la materia gris en cada plano e incluso su desaparición en algunas zonas.

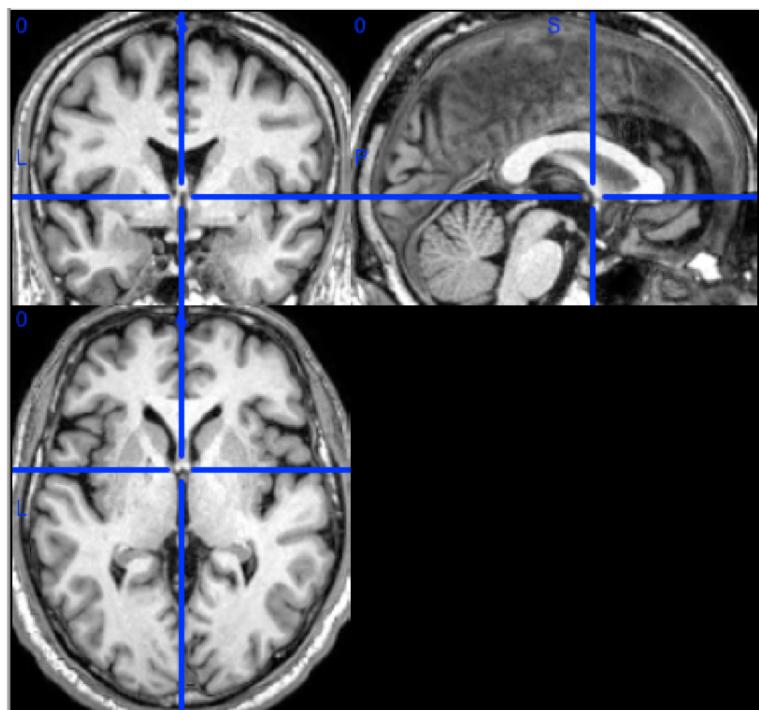


Figura 2.7: MRI de un sujeto sano

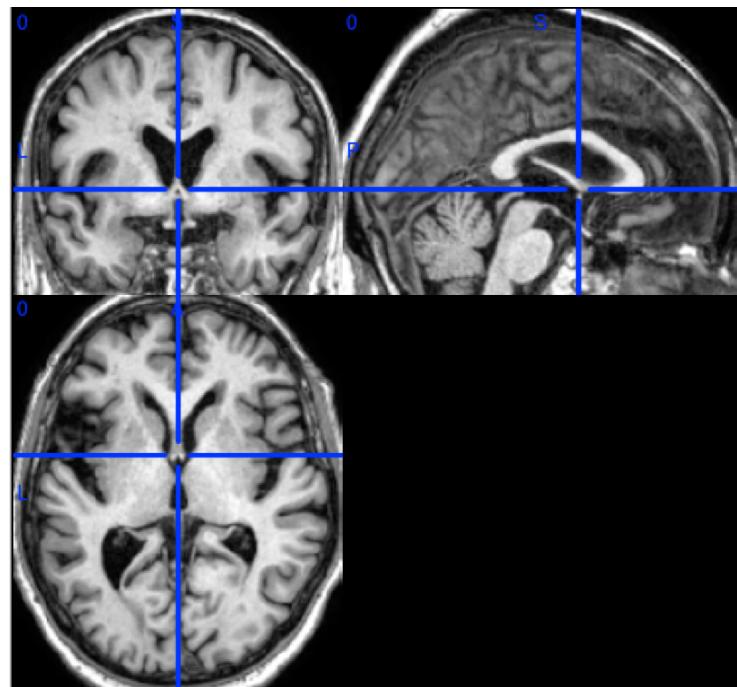


Figura 2.8: MRI de un sujeto enfermo de PD. Véase la pérdida de materia gris

3. Conceptos y herramientas matemáticas

En la presente sección discutimos los diferentes conceptos y herramientas utilizadas en este proyecto. Más allá de utilizar la multitud de técnicas que existen dentro de la ciencia de datos aplicada a la biomedicina, mi intención es profundizar desde un punto de visto matemático sus fundamentos y las garantías de su uso. Primero, introduzco la imagen de resonancia magnética (MRI) y la normalización aplicada a las imágenes para los distintos experimentos. A continuación, trataré el concepto de característica en ciencia de datos y me detendré en las herramientas utilizadas para la extracción (transformada discreta Wavelets 2D) y selección (PCA) de las mismas. Por último, abordamos el tema de la clasificación, explicando el algoritmo SVM y las optimizaciones propuestas para la mejora del rendimiento, interpretabilidad y validez de la investigación.

3.1. Resonancia magnética nuclear

Antes de trabajar con imágenes MRI, es necesario tener conocimiento sobre cómo se obtienen, así que introducimos los conceptos más importantes de la resonancia magnética nuclear (NMR [16]). Se trata de un método de imagen no invasivo que tiene como objetivo obtener información sobre la estructura y composición de un material. Es muy utilizado en medicina para recabar información sobre tejidos y observar alteraciones o degradaciones de los mismos.

NMR es un fenómeno físico basado en las propiedades cuántico-mecánicas del núcleo atómico. Primero, el *spin* (momento angular intrínseco [17]) de una partícula es un vector asociado al momento magnético de la misma. Tiene una dirección (el eje del *spin*) y un sentido. Cuando dos o más partículas tienen *spin* opuestos están pareadas, la suma de sus momentos es cero y, por tanto, no se produce manifestación alguna del *spin*. Este es el estado natural de los momentos magnéticos en el cuerpo ya que el núcleo de los átomos y sus respectivos electrones están pareados. Sin embargo, podemos encontrar en el cuerpo isótopos con *spin* distinto de cero, siendo los más comunes los de hidrógeno ya que la mayoría de los tejidos contienen agua. La manipulación del *spin* es lo que permite a la máquina de resonancia magnética encontrar las diferencias entre las orientaciones y construir una imagen. Usando campos magnéticos, los núcleos de hidrógeno se alinean magnéticamente, produciéndose este cambio en las alineaciones en un tiempo T1. En presencia de un campo magnético externo, existen dos tipos de orientaciones para el *spin* nuclear:

- Paralela, en la que el sentido del momento magnético es el mismo para la partícula y para el campo magnético externo.
- Anti-paralelo, cuando ambos dos tienen sentido contrario.

Como se muestra en la Figura 3.1, la suma de los momentos magnéticos de un grupo de núcleos de hidrógeno puede ser representado como un vector M paralelo al campo magnético externo B_0 , donde las componentes normales de los diferentes *spin* se cancelan

mutuamente. La mayoría de los *spin* adoptan la orientación paralela, por lo que la suma de los *spins* de M es paralela a B_0 .

M puede ser manipulado usando señales de radio frecuencia. Cuando se aplican, los núcleos absorben la energía y una porción de ella se emite más tarde. Esa es la señal que se detecta. Dependiendo de la intensidad de los campos magnéticos, se pueden obtener imágenes con distintas resoluciones.

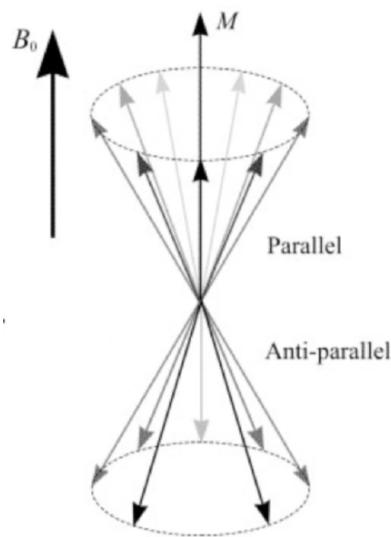


Figura 3.1: Funcionamiento de los campos magnéticos, spin y NMR

3.2. Datos de entrada: PPMI

A pesar de no ser una herramienta matemática, no puedo pasar sin mencionar, y al mismo tiempo agradecer, a *Parkinson's Progression Makers Initiative* (PPMI). PPMI es un estudio clínico de gran importancia que evalúa las zonas cerebrales de interés usando técnicas avanzadas de imagen, muestritos biológicos y comportamientos y ensayos clínicos para identificar biomarcadores de la enfermedad del Parkinson. PPMI se lleva a cabo en Estados Unidos, Europa, Israel y Australia y, en nuestro caso, evaluó el proyecto para cedernos el acceso a su base de datos (adni.loni.usc.edu), adscrita a la Universidad Sur de California.

Descargamos más de 300 imágenes 3D cerebrales en formato DICOM. DICOM es el acrónimo de *Digital Imaging and Communications in Medicine*. Es, al mismo tiempo, un protocolo de comunicación y un formato de archivo, lo que significa que almacena información médica, como imágenes MRI, junto con la información personal del paciente. El formato garantiza que todos esos datos se mantienen juntos, haciendo posible la transferencia de los mismos entre los profesionales. Tras eliminar archivos corruptos y

duplicados, obtuvimos un dataset de 242 imágenes (78 pacientes en el grupo de control y 164 enfermos de Parkinson).

3.3. Normalización

Antes de manipular o extraer información, es fundamental normalizar el conjunto de datos completo para poder comparar las imágenes. Cada imagen MRI debería representar el mismo espacio, i.e., cada región de la imagen debe describir la misma región del cerebro. Los ventrículos cerebrales deben estar situados en las mismas coordenadas para cada imagen de cada paciente.

Dado que los cerebros pueden variar mucho entre pacientes, este objetivo es difícil de conseguir ya que las herencia genética y la vida de cada paciente haga que cada uno tenga sus peculiaridades. Por tanto, debemos tener en cuenta dos ideas a cumplir:

- Normalizar la escala de grises de las imágenes para que todas estén en la misma escala.
- Redimensionar las imágenes para que todas tengan las mismas medidas.

Para completar estas dos tareas, nos remitimos a una plantilla estándar. Para la normalización, utilizamos Statistical Parametric Mapping (SPM [18]), que es un algoritmo que analiza cada *voxel* usando un test estadístico estándar. Lo utilizamos dada su gran efectividad y su librería de Matlab (a pesar de que el resto del proyecto se realiza en Python). Los valores de los *voxel* son distribuidos de acuerdo a una función de densidad, que suele provenir de las distribuciones T de Student o F. En la Figura 3.2 podemos ver un ejemplo de una imagen antes y después de la normalización. Otra de las acciones importantes llevadas a cabo sobre los datos es una segmentación que divide genera las imágenes con su contenido completo, la materia gris (C1) y la materia blanca (C2).

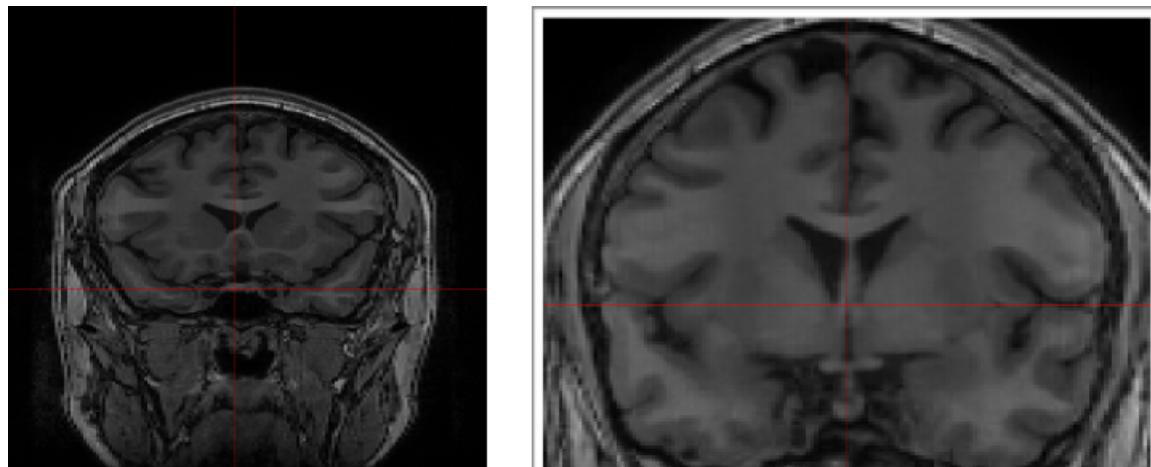


Figura 3.2: A la izquierda, imagen MRI sin normalizar. A la derecha, resultado de la normalización

3.4. Características

Una de las partes más importantes de un proyecto de inteligencia artificial es la extracción de características. Una característica puede ser entendida como una componente de un vector, que representaría los datos. Cada elemento o individuo del dataset sería representado por un vector n-dimensional (una matriz) en la que cada componente sería una característica. Esta forma de representación es muy intuitiva y flexible ya que permite la introducción y aplicación de conceptos y herramientas matemáticas.

El dataset completo sería definido, una vez que cada elemento está representado por sus características, como un conjunto de vectores de m componentes

$$F_i = [f_1, f_2, \dots, f_m]$$

En la expresión, F_i sería un único vector (un paciente) y f_i sus características. Por tanto, si tenemos un grupo de pacientes:

$$F = \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{pmatrix} = \begin{pmatrix} f_{11} & f_{12} & \dots & f_{1m} \\ f_{21} & f_{22} & \dots & f_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n1} & f_{n2} & \dots & f_{nm} \end{pmatrix}$$

3.4.1. Extracción de características

Cuando se trabaja con MRI, se tiene un número de imágenes de $157 \times 189 \times 136$ en los planos X, Y y Z respectivamente, por lo que tenemos $157*189*136 = 4.035.528$ voxels con 255 valores posibles. De esta manera encontraríamos más de mil millones de características. Necesitamos, por tanto, reducir el espacio de posibilidades, es decir, disminuir la cantidad de recursos necesaria para describir nuestro conjunto de datos.

En general, una buena estrategia para extraer características de una imagen es convertirla a un dominio diferente. Con ese objetivo, introducimos ahora dos opciones distintas: la transformada de Fourier y la transformada Wavelets (que finalmente utilizaremos). Todo el material aquí recogido proviene de multitud de textos. Son los siguientes: [19], [20], [21], [22], [23].

- **La Transformada de Fourier en $L^1(\mathbb{R}^d)$**

Dado que la transformada de Fourier es, de forma natural, una función que toma valores en los complejos, asumimos que

$$u : \mathbb{R}^d \mapsto \mathbb{C}$$

Definición 1. Sea $u \in L^1(\mathbb{R}^d)$ y $\xi \in \mathbb{R}^d$. Se define la transformada de Fourier de u en ξ como

$$(Fu)(\zeta) = \hat{u}(\xi) = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} u(x) e^{-ix\xi} dx$$

Además, podemos introducir uno de los teoremas más importantes de la transformada de Fourier: el teorema de Convolución:

Teorema de Convolución. Para $u, v \in L^1(\mathbb{R}^d)$,

$$F(u * v) = (2\pi)^{d/2} F(u)F(v)$$

Demostración.

Aplicando el teorema de Fubini, obtenemos

$$\begin{aligned} F(u * v)(\xi) &= \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} u(y)v(x-y) dy e^{-ix\xi} dx = \\ &= \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} u(y)e^{-iy\xi} v(x-y) e^{-i(x-y)\xi} dx dy \\ &= \int_{\mathbb{R}^d} u(y) e^{-iy\xi} dy F(v)(\xi) \\ &= (2\pi)^{d/2} F(u)F(v)(\xi) \end{aligned}$$

■

A pesar de este resultado, nos será de gran utilidad extender el concepto de transformada de Fourier al espacio de Hilbert $L^2(\mathbb{R}^d)$.

■ La Transformada de Fourier en $L^2(\mathbb{R}^d)$

La extensión de la transformada de Fourier al espacio $L^2(\mathbb{R}^d)$ requiere un poco más de esfuerzo. En primer lugar, definimos un “pequeño” espacio de funciones donde la transformada exhibe ciertas propiedades interesantes: el espacio de Schwartz:

Definición 2. El espacio de Schwartz se define como

$$S(\mathbb{R}^d) = \{u \in C^\infty(\mathbb{R}) | \forall \alpha, \beta \in \mathbb{N}^d : C_{\alpha, \beta}(u) = \sup_{x \in \mathbb{R}^d} |x^\alpha \frac{\partial^\beta}{\partial x^\beta} u(x)| < \infty\}$$

Una función $u \in S(\mathbb{R}^d)$ se le llama una función de Schwartz.

Grosso modo, el espacio de Schwartz contiene funciones suaves que tienden a cero más rápido que los polinomios a infinito. Puede verificarse de forma elemental que el espacio de Schwartz es un espacio vectorial. Con el objetivo de hacerlo accesible por métodos analíticos, lo dotamos de una topología. Describimos dicha topología definiendo una noción de convergencia para sucesiones de funciones.

Definición 3. Una sucesión de funciones de Schwartz u_n converge a u si y sólo si, para todo multi-índice α, β , se tiene que

$$C_{\alpha,\beta}(u_n - u) \mapsto 0 \text{ cuando } n \mapsto \infty$$

La convergencia en el espacio de Schwartz es muy restrictiva: una sucesión de funciones converge si ella y todas sus derivadas multiplicadas por monomios arbitrarios convergen uniformemente.

Los siguientes lemas serán necesarios en el desarrollo del tema:

Lema 1. El espacio de Schwartz es no vacío y cerrado con respecto a la derivación de cualquier orden y la multiplicación usual.

Demostración. Para $u \in S(\mathbb{R}^d)$, para todo multi-índice γ , tenemos

$$C_{\alpha,\beta}\left(\frac{\partial^\gamma}{\partial x^\gamma} u\right) = C_{\alpha,\beta+\gamma}(u) < \infty$$

y por tanto $\frac{\partial^\gamma}{\partial x^\gamma} u \in S(\mathbb{R}^d)$.

El hecho de que dadas $u, v \in S(\mathbb{R}^d)$, $uv \in S(\mathbb{R}^d)$ puede probarse vía la regla de Leibniz para multi-índices. ■

El espacio de Schwartz está muy relacionado con la transformada de Fourier. El siguiente lema presenta reglas de cálculo para transformadas de Fourier sobre funciones de Schwartz.

Lema 2. Sea $u \in S(\mathbb{R}^d)$, $\alpha \in \mathbb{N}^d$ un multi-índice. Se define $p^\alpha(x) = x^\alpha$. Entonces

$$F\left(\frac{\partial^\alpha u}{\partial x^\alpha}\right) = i^{|\alpha|} p^\alpha F(u)$$

$$F(p^\alpha u) = i^{|\alpha|} \frac{\partial^\alpha}{\partial x^\alpha} F(u)$$

Demostración. Comenzamos con los siguientes cálculos auxiliares:

$$\frac{\partial^\alpha}{\partial x^\alpha}(e^{-ix\xi}) = (-i)^{|\alpha|} \xi^\alpha e^{-ix\xi}$$

$$x^\alpha e^{ix\xi} = i^{|\alpha|} \frac{\partial^\alpha}{\partial \xi^\alpha}(e^{-ix\xi})$$

Aplicando integración por partes, obtenemos

$$\begin{aligned}
F\left(\frac{\partial^\alpha}{\partial x^\alpha} u\right)(\xi) &= \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \frac{\partial^\alpha}{\partial x^\alpha} u(x) e^{-ix\xi} dx \\
&= \frac{1}{(2\pi)^{d/2}} i^{|\alpha|} \xi^\alpha \int_{\mathbb{R}^d} u(x) e^{-ix\xi} dx \\
&= i^{|\alpha|} p^\alpha F u(\xi)
\end{aligned}$$

Intercambiando el orden de la integración y la derivación, llegamos a

$$\begin{aligned}
F(p^\alpha u)(\xi) &= \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} u(x) x^\alpha e^{-ix\xi} dx \\
&= \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} u(x) \frac{\partial^\alpha}{\partial \xi^\alpha} e^{-ix\xi} dx \\
&= i^{|\alpha|} \left(\frac{\partial^\alpha}{\partial \xi^\alpha} F u \right)(\xi)
\end{aligned}$$

Los argumentos anteriores son válidos dado que los integrandos son infinitamente derivables respecto a ξ e integrables respecto a x . ■

Observamos que la transformada de Fourier lleva diferenciación a multiplicación y viceversa. Esto nos lleva a que el espacio de Schwartz va a sí mismo vía la transformada de Fourier. Veámoslo:

Lema 3. Para la función $G(x) = e^{-\frac{|x|^2}{2}}$, se tiene que

$$\hat{G}(\xi) = G(\xi)$$

es decir, la función Gaussiana es una función propia de la transformada de Fourier correspondiente al valor propio uno.

Demostración

La función Gaussiana se puede escribir como un producto tensorial de Gaussianas unidimensionales $g : \mathbb{R} \mapsto \mathbb{R}$, $g(t) = \exp(-t^2/2)$ de forma que $G(x) = \prod_{k=1}^d g(x_k)$. Por el teorema de Fubini,

$$\hat{G}(\xi) = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \prod_{k=1}^d g(x_k) e^{-ix_k \xi_k} dx = \prod_{k=1}^d \hat{g}(\xi_k)$$

Para calcular la transformada de Fourier de g , tengamos en cuenta que g satisface la ecuación diferencial $g'(t) = -tg(t)$. Aplicando la transformada de Fourier en esa ecuación, por el Lema 2 obtenemos $-\omega \hat{g}(\omega) = \hat{g}'(\omega)$. De hecho, $\hat{g}(0) = \int_{\mathbb{R}} g(t) dt = 1 = g(0)$. Por tanto, las funciones g, \hat{g} satisfacen la misma ecuación diferencial con

el mismo valor inicial. Por el teorema de unicidad de soluciones en problemas de valores iniciales de Picard-Lindelöf, $g = \hat{g}$. ■

Teorema 2. La transformada de Fourier es una aplicación continua y biyectiva del espacio de Schwartz en sí mismo. Para $u \in S(\mathbb{R}^d)$, tenemos la fórmula de inversión

$$(F^{-1}Fu)(x) = \bar{\hat{u}}(x) = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \hat{u}(\xi) e^{ix\xi} d\xi = u(x)$$

Demostración. Por el lema 2, tenemos que para todo $\xi \in \mathbb{R}^d$

$$|\xi^\alpha \frac{\partial^\beta}{\partial \xi^\beta} \hat{u}(\xi)| = |F(\frac{\partial^\alpha}{\partial x^\alpha} p^\beta u)(\xi)| \leq \frac{1}{(2\pi)^{d/2}} \left\| \frac{\partial^\alpha}{\partial x^\alpha} p^\beta u \right\|_1 (*)$$

Por tanto, para $u \in S(\mathbb{R}^d)$, tenemos que $\hat{u} \in S(\mathbb{R}^d)$. Dado que la transformada de Fourier es lineal, es suficiente demostrar la continuidad en el cero. En efecto, consideramos a sucesión nula u_n en el espacio de Schwartz, es decir, cuando $n \mapsto \infty$, $C_{\alpha,\beta}(u_n) \mapsto 0$. Esto es, u_n , al igual que $\partial^\alpha p^\beta u_n$, $\forall \alpha, \beta$, converge uniformemente a cero. Esto implica que la parte derecha de $(*)$ tiende a cero. En particular, obtenemos que $C_{\alpha,\beta}(\hat{u}_n) \mapsto 0$, lo que implica que \hat{u}_n es una sucesión nula, probando la continuidad.

Para probar la fórmula de inversión, consideramos dos funciones arbitrarias $u, \phi \in S(\mathbb{R}^d)$

$$\begin{aligned} (\hat{u} * \phi)(x) &= \int_{\mathbb{R}^d} \hat{u}(y) \phi(x - y) dy = \int_{\mathbb{R}^d} \hat{u}(y) e^{ixy} \hat{\phi}(-y) dy \\ &= \int_{\mathbb{R}^d} u(y) \hat{\phi}(-x - y) dy = (u * \hat{\phi})(-x) \end{aligned}$$

Ahora elegimos ϕ para que sea una función Gaussiana reescalada:

$$\phi_\epsilon(x) = \epsilon^{-d} (D_{\epsilon^{-1}id} G)(x) = \epsilon^{-d} e^{-\frac{|x|^2}{2\epsilon^2}}$$

De estos cálculos inferimos que $\hat{\phi}_\epsilon = D_{\epsilon id} \hat{G}$ y por tanto $\hat{\phi}_\epsilon = \epsilon^{-d} D_{\epsilon^{-1} id} \hat{G}$. Por el lema 3, $\hat{G} = G$ y $\hat{\phi}_\epsilon = \phi_\epsilon$. Dado que u es en particular acotada y continua, y G es positiva con integral normalizada a uno, podemos aplicar las propiedades de la convolución y obtener que cuando $\epsilon \mapsto 0$,

$$\hat{u} * \phi_\epsilon(x) \mapsto \hat{u}(x) \text{ y } u * \phi_\epsilon(-x) \mapsto u(-x) \Rightarrow \hat{u}(x) = u(-x)$$

Por propiedades de conjugación de la transformada de Fourier, tenemos que $\bar{u} = D_{-id}\hat{u}$ y sustituyendo \hat{u} por u , obtenemos:

$$\hat{u} = D_{-id}\hat{\hat{u}} = u$$

■

Teorema 3. Hay un único operador $F : L^2(\mathbb{R}^d) \mapsto L^2(\mathbb{R}^d)$ que extiende la transformada de Fourier F a $S(\mathbb{R}^d)$ y satisface la ecuación $\|u\|_2 = \|Fu\|_2 \forall u \in L^2(\mathbb{R}^d)$. Además, F es biyectivo y su inversa F^{-1} es una extensión continua de F^{-1} en $S(\mathbb{R}^d)$

Demostración. Para $u, v \in S(\mathbb{R}^d)$, sabemos que

$$(\hat{u}, \hat{v})_2 = (u, v)_2$$

y en particular $\|u\|_2 = \|Fu\|_2$. Por tanto, la transformada de Fourier es una isometría definida en un subconjunto denso de $L^2(\mathbb{R}^d)$. Existe una única extensión continua en todo el espacio. Debido a la simetría entre F y F^{-1} , un argumento análogo demuestra el recíproco.

■

La propiedad de isometría $\|u\|_2 = \|Fu\|_2$ también implica que

$$(u, v)_2 = (Fu, Fv)_2$$

que se conoce como la fórmula de Plancherel.

■ Transformada Wavelet

En la sección anterior tratamos los fundamentos teóricos que sostienen la transformada de Fourier como herramienta para estudiar la representación de la frecuencia de una señal o imagen. Sin embargo, la información relacionada con la localización no es codificada de una forma plausible. En particular, una alteración local de una señal o una imagen da lugar a una modificación global de toda la transformada de Fourier. En otras palabras, la transformada de Fourier es una transformación global en el sentido de que $u(\hat{\xi})$ depende de todos los valores de u . En ciertas circunstancias, las transformaciones locales son deseables. Antes de introducir nuestra herramienta fundamental en la extracción de características, la transformada Wavelet, presento una nueva alternativa para estudiar información “local” en la frecuencia: la transformada de Fourier de tiempo reducido o transformada “ventana” de Fourier ([24]):

Definición 4. Sean $u, g \in L^2(\mathbb{R}^d)$. La transformada de Fourier de tiempo reducido de u con una función ventana g se define como

$$(G_g u)(\xi, t) = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} u(x) \overline{g(x-t)} e^{-ix\xi} dx$$

Esta transformada depende de un parámetro de frecuencia ξ y de un parámetro espacial t y existen muchas formas de representarla:

$$\begin{aligned}(G_g u)(\xi, t) &= F(u T_{-t} \bar{g})(\xi) \\ &= \frac{1}{(2\pi)^{d/2}} (u, M_\xi T_{-t} g)_2 \\ &= \frac{1}{(2\pi)^{d/2}} (M_{-\xi} u * D_{-id} \bar{g})(t)\end{aligned}$$

La primera alternativa explica el nombre de “ventana”: a través de la multiplicación por la función g , u es localizada previa la transformada de Fourier. Nótese que la transformada “ventana” de Fourier es una función de $2d$ variables: $(G_g u) : \mathbb{R}^{2d} \mapsto \mathbb{C}$.

Lema 4. Sean $u, v, g \in L^2(\mathbb{R}^{2d})$. Entonces $(G_g u) \in L^2(\mathbb{R}^{2d})$ y

$$(G_g u, G_g v)_{L^2(\mathbb{R}^{2d})} = \|g\|_2^2 (u, v)_2$$

Demostración. Para probar la igualdad entre productos escalares, usamos la isometría de la transformada de Fourier y la fórmula de Plancherel. Con F_t denotamos la transformada de Fourier respecto de t , utilizamos una de las alternativas de representación de la transformada ventana de Fourier y el teorema de convolución para obtener

$$\begin{aligned}F_t(G_g u(\xi, \Delta))(\omega) &= F_t((2\pi)^{d/2} (M_{-\xi} u * D_{-id} \bar{g}))(\omega) \\ &= F(M_{-\xi} u)(\omega) (F D_{-id} \bar{g})(\omega) \\ &= \hat{u}(\omega + \xi) \bar{\hat{g}}(\omega)\end{aligned}$$

Obtenemos el resultado con el siguiente cálculo:

$$\begin{aligned}(G_g u, G_g v)_{L^2(\mathbb{R}^{2d})} &= (F_t(G_g u), F_t(G_g v))_{L^2(\mathbb{R}^{2d})} \\ &= \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \hat{u}(\omega + \xi) \bar{\hat{g}}(\omega) \bar{\hat{v}}(\omega + \xi) \hat{g}(\omega) d\xi d\omega \\ &= \int_{\mathbb{R}^d} |\hat{g}(\omega)|^2 \int_{\mathbb{R}^d} \hat{u}(\omega + \xi) \bar{\hat{v}}(\omega + \xi) d\xi d\omega \\ &= \|\hat{g}\|_2^2 (\hat{u}, \hat{v})_2 \\ &= \|\hat{g}\|_2^2 (u, v)_2\end{aligned}$$

■

Vemos entonces que la transformada de Fourier de tiempo reducido es una isometría, por lo que podemos calcular la fórmula de inversión:

Corolario 1. Para $u, g \in L^2(\mathbb{R}^d)$ con $\|g\|_2 = 1$, tenemos que la fórmula de inversión es

$$u(x) = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} G_g u(\xi, t) g(x - t) e^{ix\xi} d\xi dt \text{ para casi todo } x.$$

Demostración. Dado que g está normalizada, G_g es una isometría, por lo que sólo nos queda calcular el operador adjunto. Para $u \in L^2(\mathbb{R}^d)$ y $F \in L^2(\mathbb{R}^{2d})$, tenemos que

$$\begin{aligned} (u, G_g^* F)_{L^2(\mathbb{R}^d)} &= (G_g u, F)_{L^2(\mathbb{R}^{2d})} \\ &= \int_{\mathbb{R}^{2d}} G_g u(\xi, t) \overline{F(\xi, t)} d\xi dt \\ &= \int_{\mathbb{R}^{2d}} \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^{2d}} u(x) \overline{g(x-t)} e^{-ix\xi} dx \overline{F(\xi, t)} d\xi dt \\ &= \int_{\mathbb{R}^d} u(x) \overline{\frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^{2d}} F(\xi, t) e^{ix\xi} g(x-t) d\xi dt} dx, \end{aligned}$$

lo que implica

$$G_g^* F(x) = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^{2d}} F(\xi, t) e^{ix\xi} g(x-t) d\xi dt$$

■

La transformada ventana de Fourier no se aplica en procesamiento de imágenes generalmente. Hay varios motivos: en primer lugar, la transformación de una imagen produce una función de cuatro variables. Esto nos lleva a un gran consumo de memoria y perdemos gran parte de la capacidad de visualización. Por otra parte, la discretización de esta transformada no es para nada trivial y no hay un análogo directo para series de Fourier o transformada discreta. Para más información, véase ([25]).

- Transformada Wavelet continua

Nos centramos a partir de ahora en la herramienta principal en cuanto a la extracción de características se refiere. Mientras que la transformada de tiempo reducido de Fourier usa una ventana fija para localizar la función de interés, la transformada wavelet utiliza funciones variando sus anchuras (Figura 3.3).

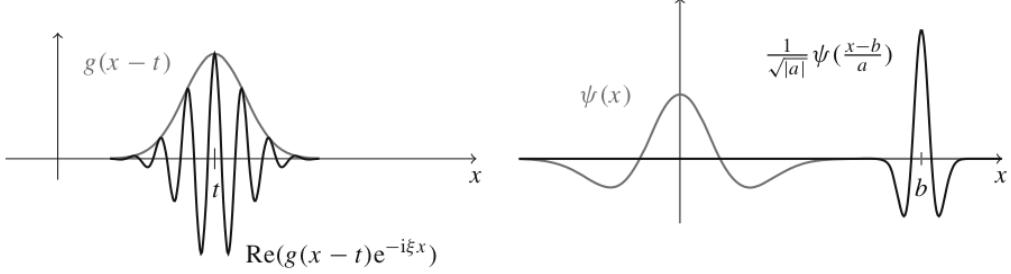


Figura 3.3: Transformada ventana (izquierda) y transformada wavelet (derecha)

En el caso de que nos encontremos en dimensión mayor que uno (nuestro caso), la transformada se puede definir de varias formas. Nos ocupamos primero de la transformada unidimensional:

Definición 5. Sean $u, \psi \in L^2(\mathbb{R}, \mathbb{R})$. Para $b \in \mathbb{R}, a > 0$ la transformada wavelet de u con ψ se define como

$$L_\psi u(a, b) = \int_{\mathbb{R}} u(x) \frac{1}{\sqrt{a}} \psi\left(\frac{x-b}{a}\right) dx$$

La transformada wavelet depende del parámetro espacial b y el parámetro de escala a . Como en casos anteriores, podemos representarla de múltiples formas:

$$\begin{aligned} L_\psi(a, b) &= \frac{1}{\sqrt{a}} (u, T_{-b} D_{1/a} \psi)_{L^2(\mathbb{R})} \\ &= \frac{1}{\sqrt{a}} (u * D_{-1/a} \psi)(b) \end{aligned}$$

Como la transformada de Fourier de tiempo reducido, la transformada wavelet posee cierta isometría, sin embargo, lo hace respecto de una medida ponderada. Introduzco el siguiente espacio:

$$L^2\left([0, \infty[\times \mathbb{R}, \frac{dadb}{a^2}\right) = \{F : [0, \infty[\times \mathbb{R} \mapsto \mathbb{R} : \int_{\mathbb{R}} \int_{\mathbb{R}^+} |F(a, b)|^2 \frac{dadb}{a^2} < \infty\}$$

El producto escalar en este espacio es

$$(F, G)_{L^2\left([0, \infty[\times \mathbb{R}, \frac{dadb}{a^2}\right)} = \int_{\mathbb{R}} \int_{\mathbb{R}^+} F(a, b) G(a, b) \frac{dadb}{a^2}$$

Teorema 4. Sea $u, \psi \in L^2(\mathbb{R})$ con

$$0 < c_\psi = 2\pi \int_{\mathbb{R}^+} \frac{|\hat{\xi}(\xi)|^2}{\xi} d\xi < \infty$$

Entonces

$$L_\psi : L^2(\mathbb{R}) \mapsto L^2\left([0, \infty[\times \mathbb{R}, \frac{dadb}{a^2}\right)$$

es una función lineal y se tiene que

$$(L_\psi u, L_\psi v)_{L^2\left([0, \infty[\times \mathbb{R}, \frac{dadb}{a^2}\right)} = c_\psi(u, v)_{L^2\mathbb{R}}$$

Demostración. Usamos el producto escalar de la transformada wavelet, las propiedades de la transformada de Fourier y la fórmula de Plancherel para obtener

$$\begin{aligned} L_\psi u(a, b) &= \frac{1}{\sqrt{a}}(u, T_{-b}D_{1/a}\psi)_{L^2(\mathbb{R})} \\ &= \frac{1}{\sqrt{a}}(\hat{u}, F(T_{-b}D_{1/a}\psi))_{L^2(\mathbb{R})} \\ &= \frac{1}{\sqrt{a}}(\hat{u}, aM_{-b}D_a\hat{\psi})_{L^2(\mathbb{R})} \\ &= \sqrt{a} \int_{\mathbb{R}} \hat{u}(\psi) e^{ib\psi} \overline{\hat{\psi}(a\xi)} d\xi \\ &= \sqrt{a2\pi} F^{-1}(\hat{u} \overline{D_a\hat{\psi}})(b) \end{aligned}$$

Ahora, calculamos lo siguiente:

$$\begin{aligned} (L_\psi u, L_\psi v)_{L^2\left([0, \infty[\times \mathbb{R}, \frac{dadb}{a^2}\right)} &= \int_{\mathbb{R}} \int_{\mathbb{R}^+} L_\psi u(a, b) L_\psi v(a, b) \frac{dadb}{a^2} \\ &= 2\pi \int_{\mathbb{R}^+} \int_{\mathbb{R}} a F^{-1}(\hat{u} \overline{D_a\hat{\psi}})(b) \overline{F^{-1}(\hat{v} \overline{D_a\hat{\psi}})(b)} db \frac{da}{a^2} \\ &= 2\pi \int_{\mathbb{R}^+} \int_{\mathbb{R}} a \hat{u}(\xi) \overline{\hat{\psi}(a\xi)} \overline{\hat{v}(\xi)} \overline{\hat{\psi}(a\xi)} da \\ &= 2\pi \int_{\mathbb{R}} \hat{u}(\xi) \overline{\hat{v}(\xi)} \int_{\mathbb{R}^+} \frac{|\hat{\psi}(a\xi)|^2}{a} dad\xi \end{aligned}$$

Un cambio de variable y $|\hat{\psi}(-\xi)| = |\hat{\psi}(\xi)|$ lleva

$$\int_{\mathbb{R}^+} \frac{|\hat{\psi}(a\xi)|^2}{a} da = \int_{\mathbb{R}^+} \frac{|\hat{\psi}(a|\xi|)|^2}{a} da = \int_{\mathbb{R}^+} \frac{|\hat{\psi}(\omega)|^2}{\omega} d\omega = \frac{c_\psi}{2\pi}$$

Aplicando la fórmula de Plancherel obtenemos lo que buscábamos. ■

Que $c_\psi < \infty$ asegura que L_ψ es una función continua y $c_\psi > 0$ garantiza la existencia de la inversa de L_ψ .

Definición 6 La condición

$$0 < c_\psi = 2\pi \int_{\mathbb{R}^+} \frac{|\hat{\psi}(\xi)|^2}{\xi} d\xi < \infty$$

se llama la condición de admisibilidad y la función ψ que la satisface se le llama *wavelets*.

La condición de admisibilidad dice que, alrededor de cero, la transformada de Fourier de una wavelet tiende suficientemente rápido a cero, $\hat{\psi}(0) = 0$.

Corolario 2 Sea $u, \psi \in L^2(\mathbb{R})$ y $c_\psi = 1$. Entonces

$$u(x) = \int_{\mathbb{R}} \int_{\mathbb{R}^+} L_\psi u(a, b) \frac{1}{\sqrt{a}} \psi\left(\frac{x-b}{a}\right) \frac{dadb}{a^2}$$

Demostración. Calculamos el adjunto de la transformada Wavelet. Para $u \in L^2(\mathbb{R})$, y $F \in L^2([0, \infty[\times \mathbb{R}, \frac{dadb}{a^2})$,

$$\begin{aligned} (L_\psi u, F)_{L^2([0, \infty[\times \mathbb{R}, \frac{dadb}{a^2})} &= \int_{\mathbb{R}} \int_{\mathbb{R}^+} \int_{\mathbb{R}} u(x) \frac{1}{\sqrt{a}} \psi\left(\frac{x-b}{a}\right) dx F(a, b) \frac{dadb}{a^2} \\ &= \int_{\mathbb{R}} u(x) \int_{\mathbb{R}} \int_{\mathbb{R}^+} F(a, b) \frac{1}{\sqrt{a}} \psi\left(\frac{x-b}{a}\right) \frac{dadb}{a^2} dx \end{aligned}$$

Esto implica que

$$L_\psi^* F(x) = \int_{\mathbb{R}} \int_{\mathbb{R}^+} F(a, b) \frac{1}{\sqrt{a}} \psi\left(\frac{x-b}{a}\right) \frac{dadb}{a^2}$$

como queríamos demostrar. ■

Presento ahora la wavelet más sencilla:

Haar wavelet:

$$\psi(x) = \begin{cases} 1 & \text{si } 0 \leq x < 0,5 \\ -1 & \text{si } 0,5 \leq x < 1 \\ 0 & \text{en otro caso} \end{cases}$$

Es discontinua pero de soporte compacto:

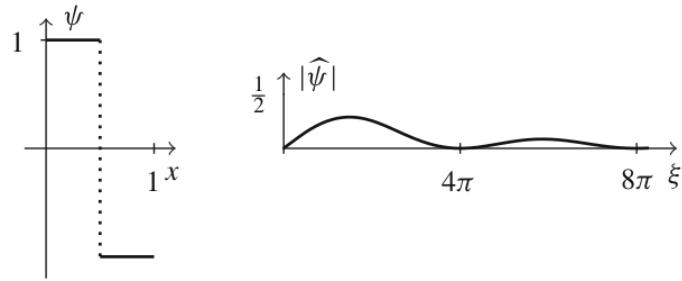


Figura 3.4: Wavelet Haar

- Transformada discreta Wavelet

La transformada continua Wavelet es una representación redundante. La pregunta que surge es si sería suficiente con conocer la transformada wavelet de una función en un subconjunto de $[0, \infty[\times \mathbb{R}$. Este es el caso para algunos subconjuntos discretos. Veamos que bajo ciertas condiciones, las funciones

$$\{\psi_{j,k}(x) = 2^{-j/2}\psi(2^{-j}x - k) | j, k \in \mathbb{Z}\}$$

forman una base ortonormal de $L^2(\mathbb{R})$. Para demostrarlo, primero necesitamos introducir ciertos conceptos. El primo de ellos es la noción de serie wavelet y transformada discreta wavelet, a saber, lo conocido como “análisis multiescala”.

Definición 7. Una sucesión $(V_j)_{j \in \mathbb{Z}}$ de subconjuntos cerrados de $L^2(\mathbb{R})$ es llamado un análisis multiescala si satisface las siguientes condiciones:

- Invarianza por traslaciones:

$$\forall j, k \in \mathbb{Z}, u_j \in V_j \Leftrightarrow T_{2^j k} u \in V_j$$

- Inclusión:

$$\forall j \in \mathbb{Z}, V_{j+1} \subset V_j$$

- Escala:

$$\forall j \in \mathbb{Z}, u \in V_j \Leftrightarrow D_{1/2} u \in V_{j+1}$$

- Intersección trivial:

$$\bigcap_{j \in \mathbb{Z}} V_j = \{0\}$$

- Completitud:

$$\overline{\bigcup_{j \in \mathbb{Z}} V_j} = L^2(\mathbb{R})$$

- Base ortonormal: Existe una función $\phi \in V_0$ tal que las funciones $\{T_k \phi | k \in \mathbb{Z}\}$ forman una base ortonormal de V_0 . La función ϕ es llamada un generador o una función de escala del análisis multiescala.

Hagamos algunas puntualizaciones sobre esta definición. Los espacios V_j son invariantes por traslación respecto de las traslaciones diádicas 2^j . Además, están encajadas entre sí y se hacen más pequeñas conforme crece j . Si notamos P_{V_j} como la proyección ortogonal sobre V_j , entonces para todo u

$$\lim_{j \rightarrow \infty} P_{V_j} u = 0, \quad \lim_{j \rightarrow -\infty} P_{V_j} u = 0$$

Definición 8. Sea $(V_j)_{j \in \mathbb{Z}}$ un análisis multiescala. Sean W_j los espacios definidos como los complementos ortogonales de V_j, V_{j-1}

$$V_{j-1} = V_j \oplus W_j, \quad V_j \perp W_j$$

El espacio V_j es llamado el espacio aproximación a la escala j ; el espacio W_j es llamado el espacio wavalet a la escala j .

La definición de W_j directamente implica

$$V_j = \bigoplus_{m \in \mathbb{Z}} W_m$$

y debido a la completitud de V_j ,

$$L^2(\mathbb{R}) = \bigoplus_{m \in \mathbb{Z}} W_m$$

Además, tenemos que $P_{V_{j-1}} = P_{V_j} + P_{W_j}$ y por tanto

$$P_{W_j} = P_{V_{j-1}} - P_{V_j}$$

Podemos ahora representar cada $u \in L^2(\mathbb{R})$ a través de los espacios V_j, W_j

$$u = \sum_{j \in \mathbb{Z}} P_{W_j} u = P_{V_m} u + \sum_{j \leq m} P_{W_j} u$$

Estas ecuaciones justifican el nombre de análisis multiescala: los espacios V_j permiten una aproximación sistemática de funciones a diferentes escalas.

Teorema 5. Sea (V_j) un análisis multiescala con generador ϕ tal que ϕ satisface

$$\phi(x) = \sum_{j \in \mathbb{Z}} h_k \sqrt{2} \phi(2x - k)$$

con una sucesión (h_k) . Más aún, sea $\psi \in V_{-1}$ definido como

$$\psi(x) = \sqrt{2} \sum_{k \in \mathbb{Z}} (-1)^k h_{1-k} \phi(2x - k)$$

Entonces:

1. El conjunto $\{\psi_{j,k} : k \in \mathbb{Z}\}$ es una base ortonormal de W_j .
2. El conjunto $\{\psi_{j,k} : k \in \mathbb{Z}\}$ es una base ortonormal de $L^2(\mathbb{R})$.
3. La función ψ es una wavelet con $c_\psi = 2 \log 2$.

Demostración. Primero vemos que para todo $k \in \mathbb{Z}$, tenemos que

$$\begin{aligned} (\psi, \phi_{k,0}) &= 0, \\ (\psi, \psi_{k,0}) &= \delta_{0,k} \end{aligned}$$

La primera ecuación implica que $\psi \perp V_0$, y por tanto tenemos que $\psi \in W_0$. La segunda ecuación implica la ortonormalidad de las traslaciones de ψ .

Ahora demostramos que el sistema $\{\psi_{k,0} | k \in \mathbb{Z}\}$ es completo en W_0 . Dado que $V_{-1} = V_0 \oplus W_0$, es equivalente demostrar que el sistema $\{\phi_{k,0} | k \in \mathbb{Z}\}$ es completo en V_{-1} . Esto último lo obtenemos viendo que $\phi_{-1,0}$ se puede representar como $\{\phi_{k,0}, \psi_{k,0} | k \in \mathbb{Z}\}$. Para ello, calculamos lo siguiente:

$$\begin{aligned} &\sum_{k \in \mathbb{Z}} |(\phi_{-1,0}, \psi_{k,0})|^2 + |(\phi_{-1,0}, \phi_{k,0})|^2 \\ &= \sum_{k \in \mathbb{Z}} \left| \sum_{l \in \mathbb{Z}} h_l (\phi_{-1,0}, \phi_{-1,l+2k}) \right|^2 + \left| \sum_{l \in \mathbb{Z}} (-1)^l h_{1-l} (\phi_{-1,0}, \phi_{-1,l+2k}) \right|^2 \\ &= \sum_{k \in \mathbb{Z}} h_{-2k}^2 + h_{1+2k}^2 = \sum_{k \in \mathbb{Z}} h_k^2 \end{aligned}$$

Se puede ver que $\sum_{k \in \mathbb{Z}} h_k^2 = 1$, y debido que $\|\phi_{-1,0}\| = 1$, se sigue que el sistema $\{\phi_{k,0}, \psi_{k,0} | k \in \mathbb{Z}\}$ es completo en V_{-1} .

Para el tercer punto, hago referencia a [26]

■

Por último, desarollo la transformada wavelet discreta bidimensional, la herramienta definitiva para nuestro estudio 2D de las MRI cerebrales:

■ La Transformada Wavelet Discreta Bidimensional

Basándonos en una base wavelet ortonormal $\{\psi_{j,k} | j, k \in \mathbb{Z}\}$ de $L^2(\mathbb{R})$, podemos construir una base ortonormal de $L^2(\mathbb{R}^2)$ a través de todos los productos tensoriales. Las funciones

$$(x_1, x_2) \mapsto \psi_{j_1, k_1}(x_1), \psi_{j_2, k_2}(x_2), \quad j_i, k_i \in \mathbb{Z}$$

forman una base ortonormal de $L^2(\mathbb{R}^2)$.

En el mismo sentido, podemos constituir un análisis multiescala de $L^2(\mathbb{R}^2)$: para un análisis multiescala (V_j) de $L^2(\mathbb{R}^2)$, creamos los espacios

$$V_j^2 = V_j \otimes V_j \subset L^2(\mathbb{R}^2)$$

que están definidos dado que las funciones

$$\Phi_{j,k} : (x_1, x_2) \mapsto \phi_{j,k_1}(x_1)\phi_{j,k_2}(x_2), k = (k_1, k_2) \in \mathbb{Z}^2$$

forman una base ortonormal de V_j^2 . Esta construcción también es llamada un producto tensorial para espacios de Hilbert separables ([27]).

En el caso bidimensional, los espacios wavelet, es decir, los complementos ortogonales de V_j^2 en V_{j-1}^2 tienen una estructura más compleja. Definimos el espacio wavelet W_j^2

$$V_{j-1}^2 = V_j^2 \oplus W_j^2$$

donde el superíndice 2 significa en un caso un producto tensorial y en el otro representa un nombre. Por otra parte, $V_{j-1} = V_j \oplus W_j$, de lo que obtenemos

$$\begin{aligned} V_{j-1}^2 &= (V_j \oplus W_j) \otimes (V_j \oplus W_j) \\ &= (V_j \otimes V_j) \oplus (V_j \otimes W_j) \oplus (W_j \otimes V_j) \oplus (W_j \otimes W_j) \end{aligned}$$

y llamando

$$H_j^2 = V_j \otimes W_j, \quad S_j^2 = W_j \otimes V_j, \quad D_j^2 = W_j \otimes W_j$$

podemos expresar

$$V_{j-1}^2 = V_j^2 \oplus H_j^2 \oplus S_j^2 \oplus D_j^2$$

Denotando la función de escala de (V_j) como ϕ y la correspondiente wavelet como ψ , definimos tres funciones:

$$\psi^1(x_1, x_2) = \phi(x_1)\psi(x_2), \quad \psi^2(x_1, x_2) = \psi(x_1)\phi(x_2), \quad \psi^3(x_1, x_2) = \psi(x_1)\psi(x_2)$$

Para $m \in 1, 2, 3, j \in \mathbb{Z}, k \in \mathbb{Z}^2$, establecemos,

$$\psi_{j,k}^2(x_1, x_2) = 2^{-j}\psi^m(2^{-j}x_1 - k_1, 2^{-j}x_2 - k_2)$$

Se puede probar que las funciones $\{\psi_{j,k}^1 | k \in \mathbb{Z}^2\}$ forma una base ortonormal de H_j^2 , las funciones $\{\psi_{j,k}^2 | k \in \mathbb{Z}^2\}$ constituyen una base de S_j^2 y las funciones $\{\psi_{j,k}^3 | k \in \mathbb{Z}^2\}$, de D_j^2 . Por tanto

$$\{\psi_{j,k}^m | m = 1, 2, 3, k \in \mathbb{Z}^2, j \in \mathbb{Z}\}$$

constituye de forma natural una base ortonormal de $L^2(\mathbb{R}^2)$.

Observamos que los espacios wavelet W_j^2 están generados por tres wavelets: ψ^1, ψ^2 y ψ^3 juntas con sus escalas y traslaciones. Los espacios H_j^2 contienen los detalles horizontales en la escala j (en la dirección x_1), los espacios S_j^2 los detalles verticales (dirección x_2) y los espacios D_j^2 los detalles en diagonal:

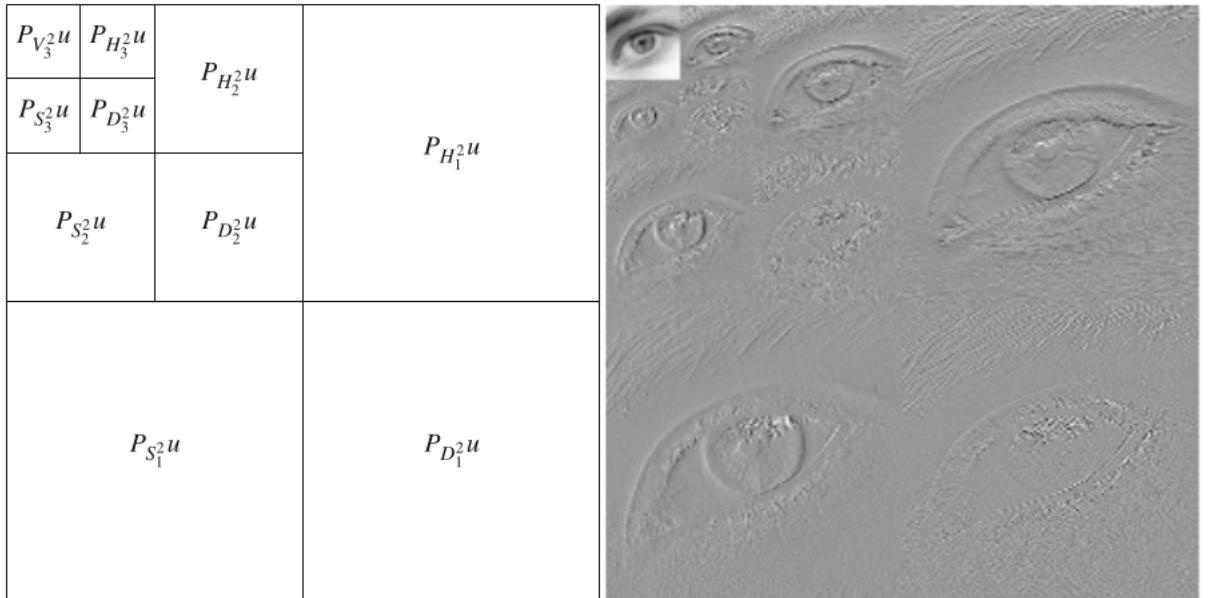


Figura 3.5: La transformada wavelet discreta 2D de una imagen a través de la wavelet Haar

Este tipo de análisis multiescala bidimensional es fácil de implementar algorítmicamente: basado en la aproximación de los coeficientes c^j , calculamos los coeficientes c^{j+1} es una escala mayor además de los tres coeficientes de detalles $d^{1,j+1}, d^{2,j+1}, d^{3,j+1}$. En la práctica, esto se consigue con la concatenación de la descomposición wavelet unidimensional a lo largo de las columnas y las filas:

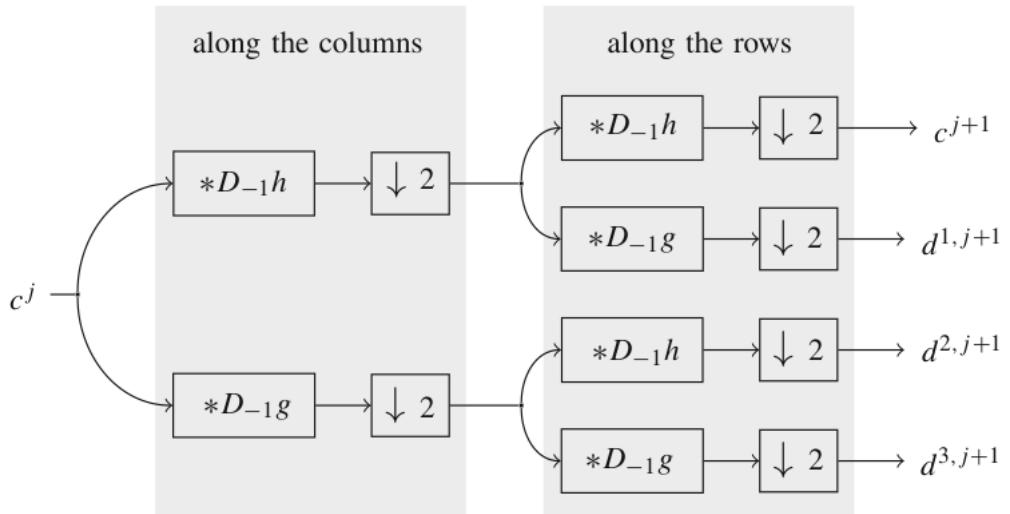


Figura 3.6: Esquema de actuación algorítmica para calcular transformada 2D

3.4.2. Selección de características

Una vez que hemos aplicado la transformada discreta wavelet 2D sobre cada una de los slices y hemos configurado nuestro dataset, nos encontramos con un conjunto de datos de enormes dimensiones, repleto de información (gran parte, probablemente, redundante o inútil en nuestro proyecto) que necesita ser estudiada. Esta basta cantidad de datos tiene dos grandes inconvenientes. El primero de ellos es que los algoritmos de aprendizaje no serán capaces de encontrar las características determinantes a la hora de clasificar instancias debido al gran ruido que tenemos (efecto Hughes). Por su parte, el segundo es un problema pragmático: los clasificadores no van a tener un rendimiento adecuado y tardarán demasiado tiempo en ejecutarse. Por tanto, es necesaria una selección de las características más importantes para así garantizar un mejor rendimiento y unas conclusiones más certeras. Existen multitud de técnicas en estadística multivariante para reducir la dimensionalidad. De entre ellas destaca Análisis de Componentes Principales (PCA) y por ello la utilizamos. PCA es una herramienta eficiente para reducir la dimensionalidad de un dataset formado un gran número de variables correladas entre sí. El objetivo es transformar el conjunto de datos en uno nuevo constituido por las variables ordenadas según el grado de importancia o varianza de las mismas. Esta técnica tiene tres efectos:

- Ortogonaliza las componentes de los vectores de entrada para que no tengan correlación entre ellas.
- Ordena el resultado de las componentes ortogonales de mayor a menor en variación o importancia.
- Elimina aquellas componentes que aportan menor variación.

Téngase en cuenta que los vectores de entrada deben estar normalizados con media cero y desviación típica uno antes de utilizar el algoritmo. La normalización se realiza tal y como se explicó en la sección anterior.

PCA encuentra una función a través de la cual incluye grandes cantidades de información en muestras analizadas. Sin embargo, como las funciones son combinaciones lineales de las características originales, es difícil (por no decir imposible) interpretar las variables resultantes de la transformación. Desde el punto de vista clínico, este hecho hace que el estudio pierda valor médico, aunque es realmente necesario debido al volumen de datos que tenemos. De manera intuitiva, dado un espacio de características n-dimensional, PCA ajusta un elipsoide de la misma dimensión que contiene un porcentaje prefijado de la varianza de los datos y cada eje se convierte en una componente principal, por lo que si un eje es pequeño, su varianza también lo será.

Desarrollo ahora el contenido matemático de Análisis de Componentes Principales ([28], [29]).

Un vector aleatorio es un vector cuyas componentes son variables aleatorias. Por tanto, la media del vector es un vector formado por las medias de cada componente. Si $X = (X_1, \dots, X_n)$ es un vector aleatorio y $\mu = (\mu_1, \dots, \mu_n)$ es su vector media, entonces

$$\mu_i = E(X_i) \quad i = 1, \dots, n$$

Sin embargo, la analogía para la varianza no es tan obvia. Definimos $var(X)$ como una matriz donde, para cada i, j , el elemento es $cov(X_i, X_j)$. Esto es lo que se llama la matriz de covarianzas de X . Esta matriz es siempre simétrica.

Si X e Y son variables aleatorias e Y es una función lineal de X , esto es, $Y = aX + b$ para alguna constante a y b , entonces

$$\begin{aligned} E(Y) &= aE(X) + b \\ var(Y) &= a^2 var(X) \\ \sigma(Y) &= |a|\sigma(X) \end{aligned}$$

Si X, Y son vectores aleatorios, entonces decimos que Y es una función lineal (afín) de X cuando $Y = AX + b$, para alguna matriz constante A y algún vector constante b . Tenemos que

$$\begin{aligned} E(Y) &= AE(X) + b \\ var(Y) &= Avar(X)A^T \end{aligned}$$

Consideremos el caso en que $Y = AX$, siendo A una matriz fila. Por tanto, AX es un escalar. Por convención, $A = a^T$, siendo a un vector columna. De la igualdad anterior tenemos que

$$0 \leq var(Y) = a^T var(X)a$$

teniendo en cuenta que la varianza de una variable aleatoria siempre es no negativa. Esta propiedad tiene un nombre. Un matriz simétrica V se dice que es semi-definida positiva si

$$a^T V a \geq 0 \quad \forall a$$

Por tanto, la matriz de covarianzas es simétrica y semi-definida positiva.

Tratamos ahora la descomposición espectral. Toda matriz simétrica tiene una descomposición espectral

$$A = ODO^T$$

donde D es diagonal y O una matriz ortogonal. Pensando en O como un cambio de sistema de coordenadas, encontramos una correspondencia con los movimientos rígidos (rotaciones) con ejes coordinados perpendiculares. La descomposición espectral puede utilizarse para determinar si una matriz es semi-definida positiva.

Proposición. Una matriz diagonal es semi-definida positiva si y solamente si todos sus elementos son no negativos.

Demostración Basta ver que

$$a^T Da = \sum_i a_i^2 d_{ii}.$$

■

En general, una matriz simétrica es semi-definida positiva si y solamente si la matriz diagonal de su descomposición espectral es semi-definida positiva, ya que

$$a^T A a = a^T O D O^T a = b^T D b$$

donde $b = O^T a$ y $a = Ob$.

Tratamos ahora los valores y vectores propios. Multiplicando a la derecha por O , tenemos que

$$AO = ODO^T O = OD$$

Si observamos este resultado por columnas $w_i \in O$, tenemos que

$$Aw_i = \lambda_i w_i$$

donde λ_i es el i -ésimo elemento de D .

Por tanto, los elementos de D son los valores propios de A y las columnas de O , sus correspondientes vectores propios.

Una vez introducidos estos sencillos conceptos, nos adentramos en Componentes Principales. Si X es un vector aleatorio de varianza finita, sea

$$\text{var}(X) = ODO^T$$

la descomposición espectral de su matriz de covarianzas.

Considero el vector aleatorio $Y = O^T X$. Entonces

$$\text{var}(Y) = O^T \text{var}(X) O = O^T O D O^T O = D$$

ya que O es ortogonal. Por tanto, Y tiene una matriz de covarianzas diagonal. Por tanto, como los elementos que están fuera de la diagonal de la matriz de covarianzas son las propias covarianzas de las variables, las componentes de Y no están correladas. Y ya que los elementos de la diagonal de la matriz de covarianzas son la varianza de cada variable y los elementos de D son los valores propios de $\text{var}(X)$, las varianzas de los componentes de Y son los valores propios de la matriz de covarianzas de X .

Los elementos de Y son las llamadas componentes principales de X . Dado que una matriz ortogonal es invertible, tenemos que $X = OY$. Esto expresa una variable aleatoria

cualquiera como una combinación lineal de variables aleatorias independientes. Este proceso de hacer la descomposición espectral de la matriz de covarianzas de X es el llamado Análisis de Componentes Principales.

PCA se usa como método de reducción de dimensionalidad. Si tomamos unos pocos componentes principales, obtenemos una explicación simple de la estructura de X a través de unas pocas variables. Deben ordenarse las componentes de Y poniendo en primer lugar las componentes más grandes (los mayores valores propios). Por último, tratamos la varianza explicada. Sea $\|\Delta\|$ la norma Euclídea de un vector, por lo que

$$\|Y\|^2 = Y^T Y = \sum_i Y_i^2$$

Como las componentes de Y son independientes, tenemos que

$$E(\|Y - v\|^2) = \sum_{i=1}^n \lambda_i$$

donde $v = E(Y)$. Dado que una transformación ortogonal es una rotación (isometría), no afecta a distancias ni longitudes. Si $\mu = E(X)$, tenemos

$$\begin{aligned} E(\|X - \mu\|^2) &= E((X - \mu)^T (X - \mu)) \\ &= E((Y - v)^T O^T (Y - v)) \\ &= E((Y - v)^T (Y - v)) \\ &= E(\|Y - v\|^2) \end{aligned}$$

Dado que $Y = O^T X$, $X = OY$ y $\mu = Ov$. De forma similar, si $\bar{\mu} = E(\bar{X})$, entonces

$$E(\|\bar{X} - \bar{\mu}\|^2) = \sum_{i=1}^k \lambda_i$$

La fracción de varianza de X explicada por las primeras k componentes principales es

$$\frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^n \lambda_i}$$

En nuestro proyecto, utilizamos siempre PCA con una varianza explicada del 95 %.

3.5. Clasificación

En esta sección expongo los distintos algoritmos que hemos usado para clasificar a los sujetos según las imágenes MRI. El primero de ellos será SVM. Más tarde, nos enfrentaremos a la necesidad de ajustar los hiperparámetros del modelo optimizando así los resultados. De igual manera, nos enfrentaremos al problema del sobre ajuste, que solucionamos con Validación Cruzada de 10 particiones. Será necesaria también la mención de las redes neuronales profundas y, finalmente, la regresión logística.

3.5.1. SVM

SVM es un clasificador (aprendizaje supervisado) muy potente introducido en 1995. Consigue resultados buenos en clasificación especialmente en diagnóstico médico. SVM se basa en el principio de minimización del riesgo estructural proveniente de la teoría del aprendizaje en estadística. Su objetivo es encontrar un hiperplano separador óptimo entre los miembros y los no miembros de una clase en un espacio de características de varias dimensiones. La entrada al algoritmo está formada por un subconjunto de características elegidas por medio de la extracción y selección de las mismas. En nuestro caso, las clases son Control (sano) y PD (enfermo de PD), por lo que dividir los pacientes sigue siendo nuestro objetivo. Veamos su fundamentación matemática ([2]):

Supongamos que disponemos de un conjunto de instancias divididas en dos clases linearmente separables. Nuestro conjunto de entrenamiento está formado por N parejas $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ con $x_i \in \mathbb{R}^p, y_i \in \{1, -1\}$. Definimos el hiperplano

$$\{x : f(x) = x^T \beta + \beta_0 = 0\}$$

donde β es un vector unitario. La función que clasifica inducida por $f(x)$ es

$$G(x) = \text{sign}[x^T \beta + \beta_0]$$

Esta última expresión representa la distancia con signo de un punto x al hiperplano $f(x) = x^T \beta + \beta_0 = 0$. Dado que las clases son separables, podemos encontrar una solución $f(x) = x^T \beta + \beta_0$ con $y_i f(x_i) > 0 \forall i$. En consecuencia, somos capaces de encontrar un hiperplano que genera el mayor *margen* entre las instancias de entrenamiento para las clases 1 y -1 (Figura 3.7)

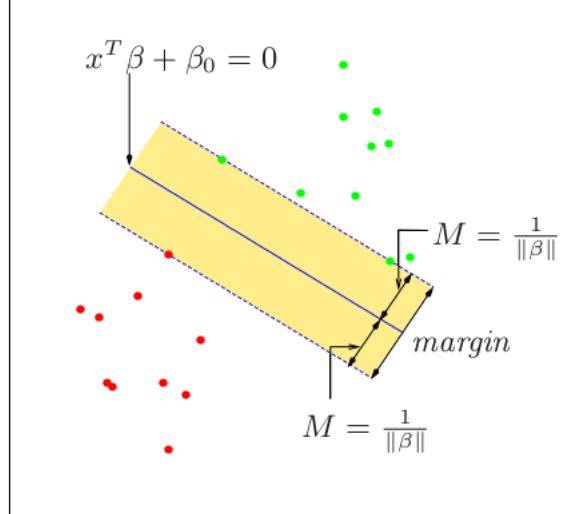


Figura 3.7: Hiperplano separador con el margen y los vectores soporte (aquellos que definen la región)

El problema de optimización es el siguiente:

$$\max_{\beta, \beta_0, \|\beta\|=1} M$$

$$\text{sujepto a } y_i(x_i^T \beta + \beta_0) \geq M, i = 1, \dots, N$$

La banda tiene una anchura de $2M$ ($M = \frac{1}{\|\beta\|}$ desde cada clase) y se llama margen. Por la definición de M , es posible reformular el problema de una forma más sencilla

$$\min_{\beta, \beta_0} \beta, \beta_0 \beta$$

$$\text{sujepto a } y_i(x_i^T \beta + \beta_0) \geq M, i = 1, \dots, N$$

Este es un problema de optimización con funciones convexas que se resuelve con los procedimientos clásicos del análisis real de varias variables.

Supongamos ahora que las clases se solapan en el espacio de características. Una forma de enfrentarnos a ese solapamiento es maximizar M permitiendo a algunos puntos estar mal clasificados (quedarse en el lado erróneo del margen). Si definimos esa anomalía a través de una variable $\xi = (\xi_1, \dots, \xi_n)$, podemos representar el problema de restricciones anterior de la siguiente manera

$$y_i(x_i^T \beta + \beta_0) \geq M(1 - \xi_i)$$

$$\forall i, \xi_i \geq 0, \sum_{i=1}^N \xi_i \leq cte.$$

La idea es la siguiente. El valor ξ_i en la restricción $y_i(x_i^T \beta + \beta_0) \geq M(1 - \xi_i)$ es la cantidad proporcional por la que la predicción $f(x_i) = x_i^T \beta + \beta_0$ está en el lado erróneo del margen. Por tanto, acotando $\sum \xi_i$, acotamos la cantidad proporcional total a las predicciones mal

situadas. Estos errores en la clasificación ocurren cuando $\xi_i > 1$, así que acotar $\sum \xi_i$ por una constante K acota superiormente el número de instancias mal clasificadas por K . Como antes, podemos eliminar la restricción sobre la norma de β , ya que $M = \frac{1}{\|\beta\|}$ y escribimos

$$\begin{aligned} & \min \|\beta\| \\ \text{sujeto a } & y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i \forall i \text{ y } \xi_i \geq 0, \sum \xi_i \leq cte. \end{aligned}$$

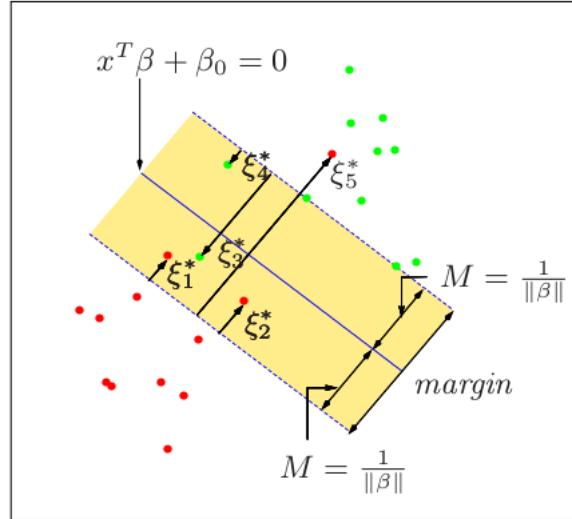


Figura 3.8: Hiperplano separador con el margen y clases solapadas

El problema anterior es cuadrático con inecuaciones lineales como restricciones, luego es un problema de optimización convexo. Para resolverlo, utilizamos los multiplicadores de Lagrange. Reescribimos la expresión de una forma más conveniente desde el punto de vista computacional

$$\begin{aligned} & \min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^N \xi_i \\ \text{sujeto a } & \xi_i \geq 0, y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i \forall i \end{aligned}$$

donde C es un parámetro que refleja el coste (para un problema donde es linealmente separable, $C = \infty$).

El lagrangiano es

$$L_p = \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i [y_i(x_i^T \beta + \beta_0) - (1 - \xi_i)] - \sum_{i=1}^N \mu_i \xi_i$$

en la que minimizamos respecto de β, β_0 y ξ_i . Derivando e igualando a cero obtenemos

$$\beta = \sum_{i=1}^N \alpha_i y_i x_i,$$

$$0 = \sum_{i=1}^N \alpha_i y_i$$

$$\alpha_i = C - \mu_i \forall i$$

además de las restricciones $\alpha_i, \mu_i, \xi_i \geq 0 \forall i$. Sustituyendo en la expresión anterior, obtenemos que

$$L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{i'=1}^N \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'}^T$$

Ahora, maximizamos L_D sujeto a que $0 \leq \alpha_i \leq C$ y $\sum_{i=1}^N \alpha_i y_i = 0$. Además de las ya presentadas, las condiciones de Karush-Kuhn-Tucker ([30]) incluyen las restricciones

$$\alpha_i [y_i(x_i^T \beta + \beta_0) - (1 - \xi_i)] = 0$$

$$\mu_i \xi_i = 0$$

$$y_i(x_i^T \beta + \beta_0) - (1 - \xi_i) \geq 0$$

para $i = 1, \dots, N$

De la igualdad correspondiente a β , vemos que la única solución tiene la forma

$$\hat{\beta} = \sum_{i=1}^N \hat{\alpha}_i y_i x_i$$

para coeficientes $\hat{\alpha}_i$ distintos de cero que cumplen las restricciones citadas. Estas observaciones son los llamados vectores soporte.

SVM y núcleos

El clasificador descrito hasta ahora encuentra los bordes de los datos en el espacio de características de forma lineal. Podemos hacer este procedimiento más flexible aumentando el espacio de características usando extensiones basadas en polinomios o splines. Generalmente, los bordes lineales consiguen una mejor separación entre las clases del conjunto de entrenamiento en espacio mayores y se traducen en bordes no lineales en el espacio original. Una vez que las funciones base $h_m(x), m = 1, \dots, M$ han sido seleccionadas, el procedimiento es igual que antes. Ajustamos el clasificador usando las características $h(x_i) = (h_1(x_i), \dots, h_M(x_i)), i = 1, \dots, N$ y producimos la función (no lineal) $\hat{f}(x) = h(x)^T \hat{\beta} + \hat{\beta}_0$. El target finalmente es, como antes, $\hat{G}(x) = sign(\hat{f}(x))$. Utilicemos esta idea para extender el concepto de SVM. Si escribimos la función lagrangiana dual

$$L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{i'=1}^N \alpha_i \alpha_{i'} y_i y_{i'} \langle h(x_i), h(x_{i'}) \rangle$$

y utilizando la expresión de f , tenemos que

$$\begin{aligned} f(x) &= h(x)^T \beta + \beta_0 \\ &= \sum_{i=1}^N \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0 \end{aligned}$$

Como antes, dado α_i, β_0 puede calcularse por medio de la ecuación

$$y_i f(x_i) = 1$$

$\forall x_i$ con $0 < \alpha_i < C$.

Las dos expresiones anteriores involucran a h a través de productos escalares. En realidad, no necesitamos conocer h si no el núcleo

$$K(x, x) = \langle h(x), h(x) \rangle$$

que calcula los productos escalares en los espacios transformados. K debe ser simétrica y semidefinida positiva. Existen tres núcleos popularmente utilizados en la literatura:

- Polinomial de grado d : $K(x, x') = (1 + \langle x, x' \rangle)^d$
- Base radial: $K(x, x') = \exp(-\gamma \|x - x'\|^2)$
- Red neuronal: $K(x, x') = \tanh(k_1 \langle x, x' \rangle + k_2)$

Finalmente, la solución tiene la expresión

$$\hat{f}(x) = \sum_{i=1}^N \hat{\alpha}_i y_i K(x, x_i) + \hat{\beta}_0$$

3.5.2. Validación cruzada con K-particiones

Validación cruzada es una técnica de validación de modelos que determina la robustez de los resultados de nuestro análisis estadístico. De forma específica, hace más robusta la fase de entrenamiento haciendo particiones k sucesivas, a su vez, de entrenamiento y validación, de forma que $k-1$ partes quedan para entrenar y una para testear el modelo. El proceso es llevado a cabo k veces reduciendo así la variabilidad. En nuestro proyecto utilizamos 10 particiones.

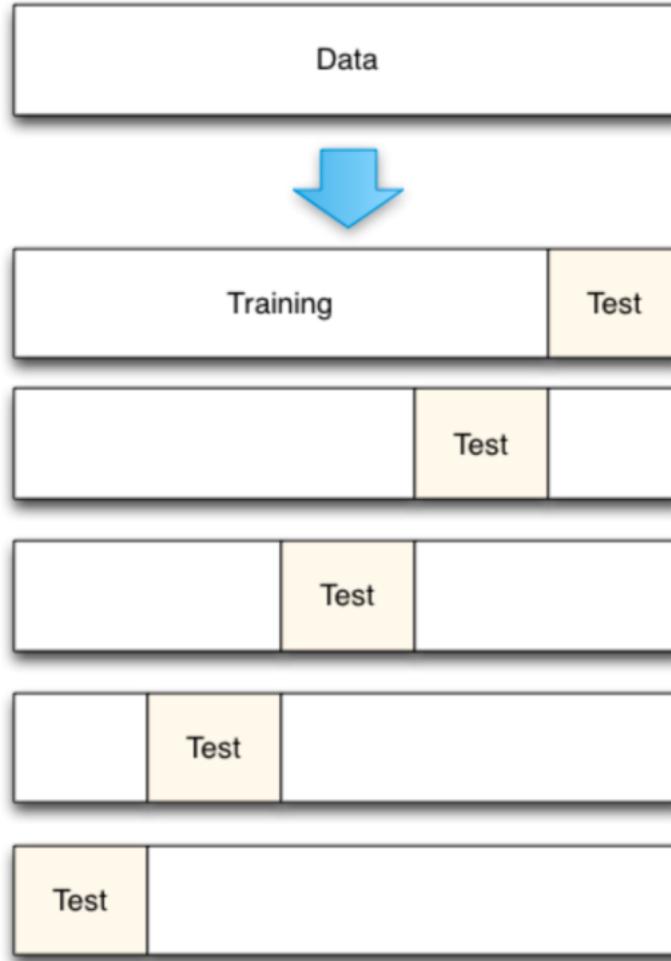


Figura 3.9: Ejemplo de validación cruzada

3.5.3. Ajuste de hiperparámetros y GridSearch

Mientras ejecutamos la validación cruzada de 10 particiones intentamos encontrar los mejores parámetros C y γ a través de un grid search. Grid Search es una técnica utilizada para la optimización de hiperparámetros, asegurando así que los modelos no sobre ajustan. Este proceso consiste en una búsqueda exhaustiva en un rango definido para la pareja (C, γ) . Aquella pareja que consiga el mayor *recall* (*accuracy* no es la mejor medida en biomedicina, puesto que lo que queremos evitar son los falsos negativos) es seleccionada para el training. Los rangos de valores son los siguientes:

- $C = \{2^i, i = -2, \dots, 11\}$
- $\gamma = \{2^i, i = -9, \dots, 3\}$

3.5.4. Regresión logística

La última pieza de nuestro clasificador, antes de explicar la arquitectura completa, es la regresión logística ([2]). Los modelos de regresión logística surgen por el deseo de modelar la probabilidad a posteriori de las K clases a través de una función lineal en x , mientras que al mismo tiempo se aseguran de que suman 1 y se mantienen en el intervalo [0,1] (es una probabilidad). el modelo tiene la forma

$$\begin{aligned} \log \frac{P(G = 1|X = x)}{P(G = K|X = x)} &= \beta_{10} + \beta_1^T x \\ \log \frac{P(G = 2|X = x)}{P(G = K|X = x)} &= \beta_{20} + \beta_2^T x \\ &\vdots \\ \log \frac{P(G = K - 1|X = x)}{P(G = K|X = x)} &= \beta_{(K-1)0} + \beta_{K-1}^T x \end{aligned}$$

A pesar de que el modelo utiliza la última clase en el denominador de cada expresión, la elección de dicho denominador es arbitraria y la estimaciones son invariantes bajo esta elección. Un simple cálculo muestra que

$$\begin{aligned} P(G = k|X = x) &= \frac{\exp(\beta_{k0} + \beta_k^T x)}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T x)} \\ P(G = K|X = x) &= \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T x)} \end{aligned}$$

con $k = 1, \dots, K - 1$ y claramente suman 1. Para enfatizar la dependencia sobre el conjunto de parámetros $\theta = \beta_{10}, \beta_1^T, \dots, \beta_{(K-1)0}, \beta_{K-1}^T$, denotamos las probabilidades $P(G = k|X = x) = p_k(x; \theta)$.

Cuando $K = 2$ (nuestro caso), el modelo especialmente simple, ya que se trata exclusivamente de una función lineal.

Ajustando un modelo de regresión logística

Los modelos de regresión logística se ajustan habitualmente con el principio de máxima verosimilitud, usando la verosimilitud condicionada a G dado X . Dado que $P(G|X)$ especifica completamente la distribución condicional, la utilización de la multinomial es adecuada. La *log-verosimilitud* para N observaciones es

$$l(\theta) = \sum_{i=1}^N \log p_{g_i}(x_i; \theta)$$

donde $p_k(x_i; \theta) = P(G = k|X = x_i)$. Nos centramos en $K = 2$, puesto es el caso que nos ocupa. Supongamos que las dos clases que tenemos se codifican como 0/1, de forma que

$y_i = 1$ cuando $g_i = 1$ y $y_i = 0$ cuando $g_i = 2$. Sea $p_1(x; \theta) = p(x; \theta), p_2(x; \theta) = 1 - p(x; \theta)$. La *log-verosimilitud* se puede escribir como

$$\begin{aligned} l(\beta) &= \sum_{i=1}^N \{y_i \log p(x_i; \beta) + (1 - y_i) \log(1 - p(x_i; \beta))\} \\ &= \sum_{i=1}^N \{y_i \beta^T x_i - \log(1 + e^{\beta^T x_i})\} \end{aligned}$$

Aquí, $\beta = \{\beta_{10}, \beta_1\}$ y asumimos que el vector de entrada x_i incluye el término constante 1. Para maximizar la *log-verosimilitud*, derivamos e igualamos a cero:

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^N x_i(y_i - p(x_i; \beta)) = 0$$

siendo $p + 1$ ecuaciones no lineales en β . Para resolverlas utilizamos el algoritmo de Newton-Raphson, que requiere el cálculo de la matriz Hessiana

$$\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} = - \sum_{i=1}^N x_i x_i^T p(x_i; \beta)(1 - p(x_i; \beta))$$

Empezando con un β^{old} , tenemos que

$$\beta^{new} = \beta^{old} - \left(\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} \right)^2 \frac{\partial l(\beta)}{\partial \beta}$$

donde las derivadas se evalúan en β^{old} . Escribimos nuestras ecuaciones en forma matricial, de forma que \mathbf{y} es el vector que contiene a las y_i , \mathbf{X} , de $N \times (p + 1)$ la matriz de x_i , \mathbf{p} el vector de probabilidades $p(x_i; \beta^{old})$ y \mathbf{W} una matriz diagonal $N \times N$ con los pesos $p(x_i; \beta^{old})(1 - p(x_i; \beta^{old}))$. Así, tenemos

$$\begin{aligned} \frac{\partial l(\beta)}{\partial \beta} &= \mathbf{X}^T (\mathbf{y} - \mathbf{p}) \\ \frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} &= -\mathbf{X}^T \mathbf{W} \mathbf{X} \end{aligned}$$

y por tanto, los pasos del método de Newton son

$$\begin{aligned} \beta^{new} &= \beta^{old} + (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{y} - \mathbf{p}) \\ &= (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} (\mathbf{X} \beta^{old} + \mathbf{W}^{-1} (\mathbf{y} - \mathbf{p})) \\ &= (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{z} \end{aligned}$$

Parece que $\beta = 0$ es un buen punto de inicio para el proceso iterativo, aunque la convergencia nunca está garantizada. Generalmente el algoritmo converge, ya que la *log-verosimilitud* es cóncava.

3.5.5. Optimización: En busca de los mejores cortes

Una vez explicadas las principales herramientas dentro de la clasificación, es momento de conectarlas y darles sentido. Nuestro objetivo es encontrar aquellos cortes (*slices*) que nos dan la información más veraz a la hora de diagnosticar la enfermedad del Parkinson. En trabajos anteriores como [14] o [15] se utilizó un algoritmo genético como optimización (NSGA-II). Yo propongo una alternativa que considero más eficiente desde un punto de vista computacional. Mi idea nace en el seno de los llamados *ensemble learner*. En particular, utilizo *stacking* de la siguiente manera. En una primera fase, utilizando solamente los 189 cortes del plano Coronal (se explicará en la siguiente sección por qué) genero 189 clasificadores, uno por slice, de manera que cada paciente pasa por todos ellos (en su slice correspondiente, especializándose en ella), aprende las características propias de las mismas y genera una predicción sobre el conjunto de test. Así, cuando se han completado los 189 procesos, contamos con un dataset nuevo repleto de las predicciones generadas por los modelos, de forma que cada fila es un paciente y cada columna un slice (por lo que la dimensión es $Num_pacientes \times 189$). Aquí interviene la segunda parte del *stacking*, en la que empleo un modelo lineal (regresión logística) para calcular los pesos de cada slice, viendo cuál es la que tiene más transcendencia a la hora de clasificar un paciente como enfermo o sano. En cada una de las fases, separo los datos en train y test, llevando a cabo una validación cruzada.

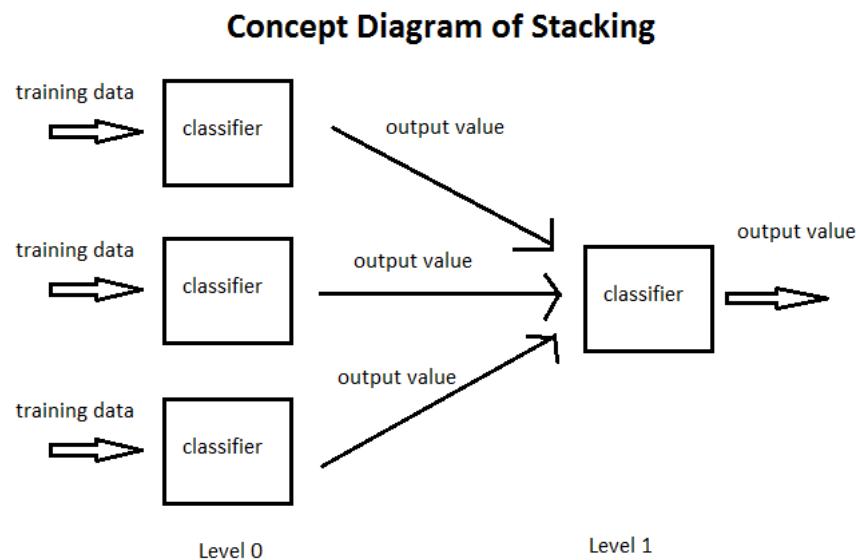


Figura 3.10: Modelo de stacking

3.5.6. Deep Learning y Stacking

Antes de describir con detalle la herramienta de deep learning que he utilizado, las Redes Neuronales Convolucionadas (CNN), introduzco qué es una Red Neuronal. Para su estudio, he realizado los cursos [31], [32], consultas en [1] y he seguido de forma fidedigna

el libro [33].

■ REDES NEURONALES

Una Red Neuronal es un sistema computacional que trata de imitar o aproximar la sinapsis que se produce entre las neuronas del cerebro humano. Matemáticamente hablando, podemos decir que estos sistemas tratan de aproximar funciones dados unos input (de hecho, son aproximadores universales). Estas redes, al igual que un cerebro humano, están compuestas por neuronas. Cada neurona posee un peso, una función de activación y está conectada con otro conjunto de neuronas:

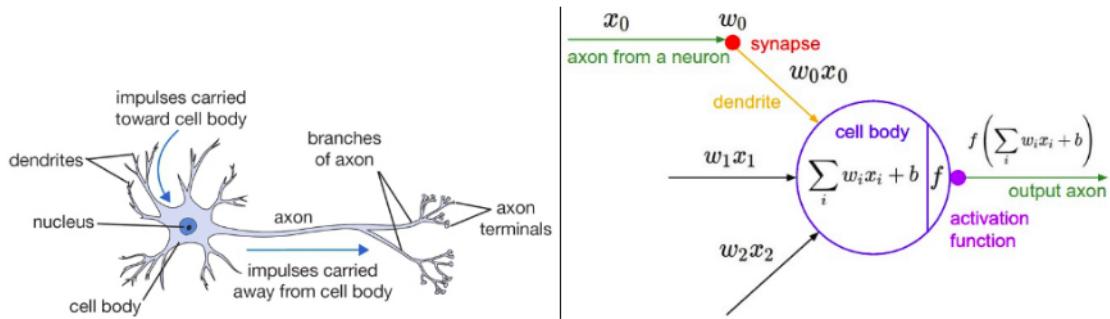


Figura 3.11: Comparación entre el cerebro humano y una red neuronal ([1])

Respecto de la topología de las redes y su conectividad, encontramos distintos casos en la literatura ([33]):

- En cadena
- En cascada
- Con saltos
- Con múltiples cabezas
- Arquitecturas especializadas

Como ejemplos de funciones de activación, presento las más significativas ([1]):

1. Función sigmoide σ , descrita por la curva $\sigma(x) = \frac{1}{1+e^{-x}}$, produciendo valores en el intervalo $[0,1]$. Son útiles para la última capa de la red para clasificaciones multi etiqueta.
2. Función ReLu (*Rectified Linear Unit*) cuya curva es $f(x) = \max(0, x)$.
3. Función *Softmax*. Normalmente interpretada como una distribución de probabilidad, tiene la siguiente expresión $\sigma(x)_i = \frac{e^{x_i}}{\sum_{k=1}^K e^{x_k}}$

Las redes neuronales suelen tener una gran cantidad de capas. Aquellas que no son las de entrada y la salida se llaman capas ocultas, y sirven para aproximar funciones más complejas.

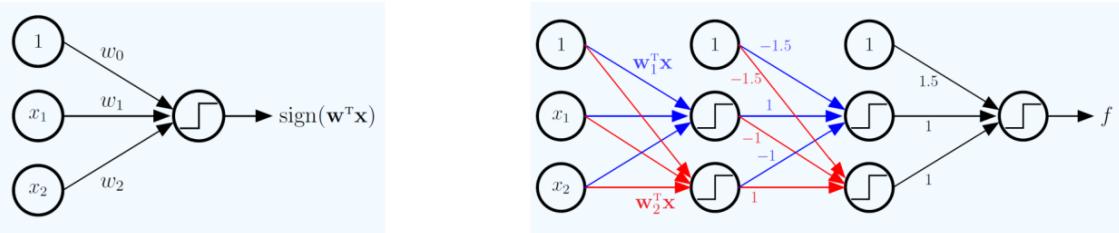


Figura 3.12: Ejemplo de red sencilla (izquierda) y red con capas ocultas (derecha) ([2])

- CNN

Una vez terminada la breve introducción sobre redes neuronales, nos introducimos en las redes neuronales convolutivas. Las redes convolutivas son las redes neuronales artificiales que se utilizan habitualmente para resolver múltiples problemas prácticos que requieren procesar imágenes. Por ejemplo, cuando la cámara frontal de un vehículo autónomo capta una señal de tráfico, debe identificar de qué señal concreta se trata. También puede interesarnos detectar qué tipo de objetos parecen en la imagen correspondiente a una escena y localizarlos dentro de la imagen (detección). Incluso podemos utilizar redes convolutivas como parte de un sistema que genera una descripción textual del contenido de una imagen, con lo que podemos indexar imágenes para realizar búsquedas por contenido en bases de datos de imágenes o sintetizar una señal de voz a partir de la descripción textual para usuarios invidentes.

¿En qué se diferencian las redes convolutivas de las redes multi capa presentadas antes? Principalmente, en que tanto sus entradas como sus salidas pueden ser estructuradas. Las redes convolutivas nos permitirán aprovechar dicha estructura para diseñar arquitecturas especializadas que resuelvan de un modo más eficiente problemas que trabajen con tipos particulares de señales.

En lugar de recibir un vector de entradas correspondientes a diferentes variables, recibiremos como entrada un vector, matriz o tensor en el que podemos explotar la relación física existente entre las diferentes entradas. EN el caso de señales unidimensionales, puede tratarse de una señal de audio en el que entradas adyacentes corresponden a muestras consecutivas en el tiempo. En el caso de señales bidimensionales, las entradas pueden corresponder a los píxeles de una imagen. También podemos tener señales bidimensionales de audio, como las provenientes de un array de micrófonos. En ocasiones, las señales de entrada puede que tengan más de dos dimensiones, como las imágenes en color, los datos volumétricos de imágenes médicas o un simple vídeo.

La salida de una red convolutiva puede ser convencional, para resolver problemas de clasificación usando una capa softmax (nuestro caso) o problemas de regresión. No obstante, habrá situaciones en las que su salida también reflejará la estructura de problemas que pretendemos resolver. Si trabajamos con imágenes, podemos construir una red convolutiva cuya salida también sea una imagen ([33]).

En lo siguiente, establecemos que nuestras entradas serán imágenes (slices del plano coronal). Estas redes tienen la particularidad de que las capas más cercanas a la imagen buscan características de bajo nivel, como es el caso de formas, texturas y colores. Mientras que las capas más cercanas a la salida buscan características semánticas de alto nivel, como personas, animales, tumores o deformaciones. Cabe destacar que basan su eficacia en la cantidad de capas dentro de su arquitectura ([1]). Conforme crece el número de capas, el rendimiento mejora aunque aumenta, de igual manera, el coste computacional.

Situémonos en nuestro caso. Nuestras imágenes tienen unas dimensiones de $157 \times 136 \times 1 = 21352$ neuronas o pesos que debemos actualizar y propagar (el 1 es por estar en blanco y negro). Como se puede imaginar, el coste es brutal. Por ello, utilizamos capas especiales que palíen este defecto, como son las capas *convolutional*, *pooling* y *dropout*.

- Capas *Convolutional*

En ellas, cada neurona de una capa posterior cuenta con un rango de visión de la capa anterior, llamado campo local receptivo, de forma que las neuronas de la capa anterior que presenten una activación adecuada, contribuirán (a través de la convolución) a la entrada de la capa posterior. El ancho y alto del campo local receptivo se denomina filtro y la cantidad de píxeles de la ventana se conoce como paso o stride. El objetivo es detectar una cierta característica en ventanas superpuestas y en distintas posiciones de la imagen. A las capas posteriores se les conoce como mapa de características. Después de cada convolución, se suele introducir una capa de activación ReLu.

- Capas *Pooling*

Debido a que la información generada por los mapas de características es ingente, es necesario resumirlos en una matriz que contenga la información más relevante. A este proceso se le llama Pooling. Cada neurona extraerá solamente el valor máximo o el valor medio de cada ventana (según el algoritmo) ignorando el resto. Así pues, el objetivo final de estas capas es reducir el número de pesos, evitando el sobre aprendizaje. [1]

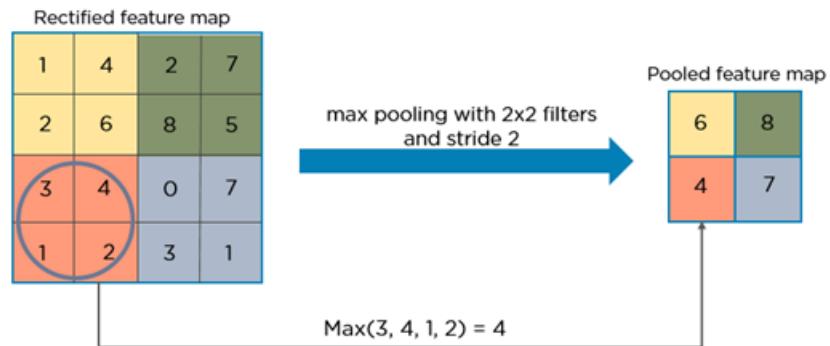


Figura 3.13: Ejemplo de pooling con máximo

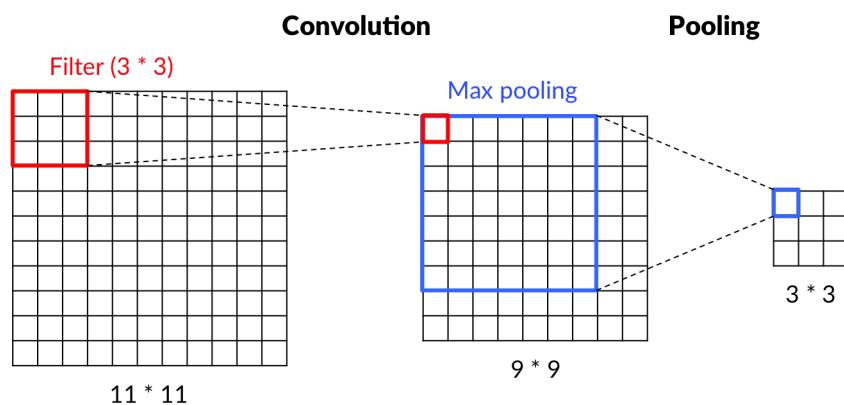


Figura 3.14: Ejemplo de convolución + pooling

- Capas *Dropout*

Una nueva medida para evitar el sobre ajuste. Se “desconectan” neuronas de forma aleatoria (llevar a cero un conjunto aleatorio de activaciones).

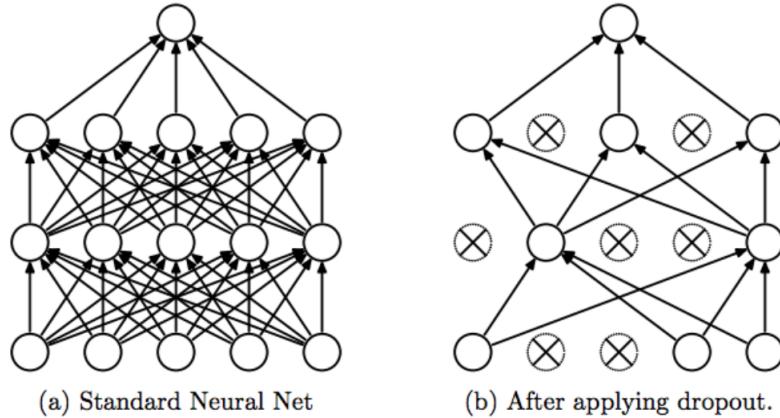


Figura 3.15: Ejemplo de dropout

Tras la introducción a redes neuronales y deep learning, relaciono la técnica *stacking* con las redes profundas convolutivas. Mi voluntad es comparar los métodos de clasificación tradicionales (en este trabajo encarnados por SVM) con las redes, por lo que sustituiré SVM por CNN para completar la optimización en la búsqueda de las mejores slices. Todo ello será explicado con detenimiento en la siguiente sección.

4. Metodología y experimentos

En la presente sección desglosó los distintos experimentos realizados, explicando sus motivaciones y resultados principales. En resumen, estos han sido los experimentos:

- El primer experimento consistió en decidir qué plano de las imágenes MRI es más importante para clasificar sujetos: X (Axial), Y (Coronal) o Z (Sagittal). Esto nos hizo reducir tiempo de cómputo y slices poco útiles.
- Tras decidir que plano utilizar, estudiamos el accuracy de cada slice y comparamos el rendimiento de la materia gris respecto al cerebro completo para mejorar el rendimiento de la clasificación.
- Primera optimización para los cortes más relevantes, en particular cortes del plano Coronal usando stacking y SVM.
- Segunda optimización con los mismos cortes usando stacking y CNN.

4.1. Experimento 1: Determinando el mejor plano

En el experimento nos enfrentamos a la cuestión de qué plano dentro de una MRI es más representativo a la hora de clasificar a los pacientes, de forma que en lo sucesivo sólo utilizaremos este plano, ahorrando así tiempo de cómputo.

Usamos 100 pacientes en este experimento, 45 en el grupo de control y 55 enfermos de PD de la base de datos PPMI, normalizamos con la herramienta SPM de Matlab y utilizamos SVM con el núcleo RBF.

4.1.1. Material y metodología

Las imágenes de los sujetos fueron, en primer lugar, segmentadas a través de la rutina de segmentación implementada en SPM12 ([18]). Tras ello, normalizamos y cambiamos el tamaño de la imagen a los bordes [-78 -112 -60; 78 76 85]. Extraemos características con la transformada discreta Wavelets 2D usando la transformada Bior3.3 hasta nivel 2. Tras ello, seleccionamos las características que explican el 95 % de la variabilidad en el conjunto de datos con PCA. Por último, el resultado de PCA fue ejecutado en SVM usando validación cruzada de 10 particiones y un conjunto de entrenamiento del 70 % de los datos.

Computamos los coeficientes wavelets 2D para cada slice y para cada paciente y llevamos a cabo la clasificación. Cada vez que entrenamos un SVM se realiza una permutación aleatoria de los pacientes para evitar el sesgo, de forma que el modelo no sobre ajusta, y realizamos 10 clasificaciones cada vez, quedándonos con la media de los resultados por cada slice.

Este proceso se lleva a cabo en cada uno de los planos de la imagen MRI: X,Y y Z. A continuación, presentamos los resultados.

4.1.2. Resultados

Queríamos comprobar qué plano tiene el mejor accuracy para así usarlo en los siguientes pasos. Analizamos los resultados de la clasificación para cada slice por separado. Llevamos a cabo esta prueba para cada tipo de plano, realizando la clasificación utilizando las mismas características en cada slice para cada sujeto.

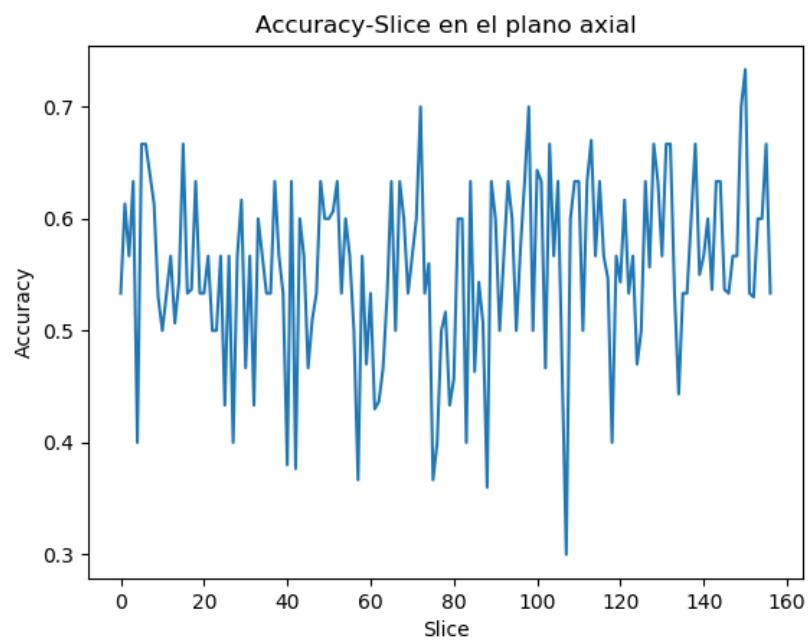


Figura 4.1: Resultados para el plano axial

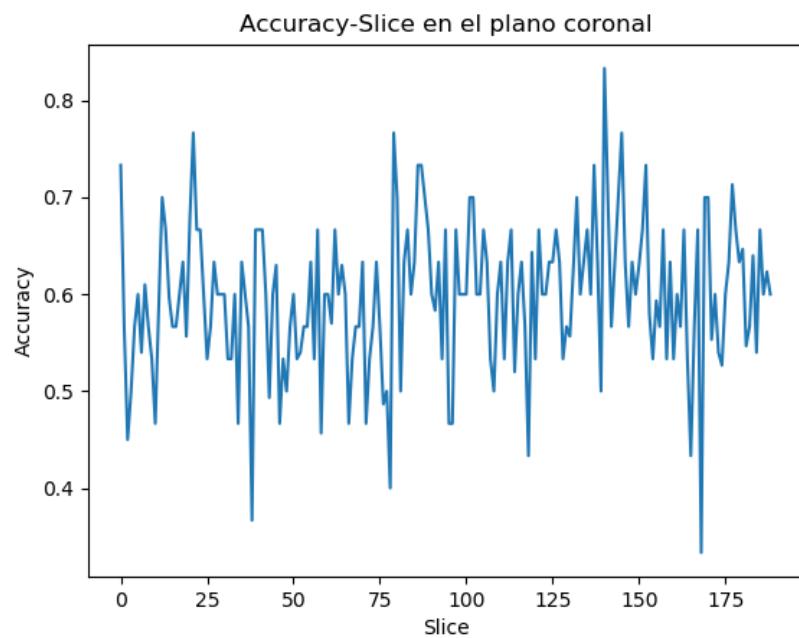


Figura 4.2: Resultados para el plano coronal

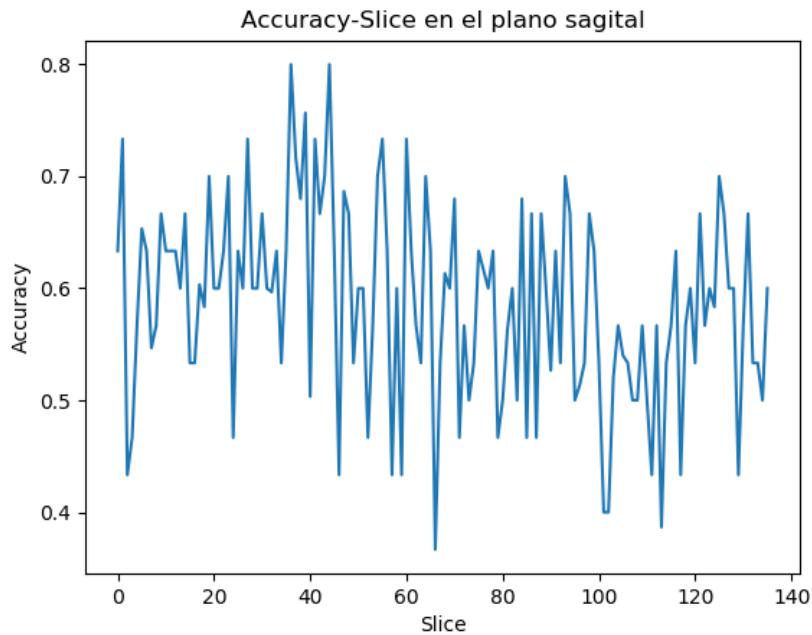


Figura 4.3: Resultados para el plano sagittal

A primera vista no parece haber mucha diferencia entre las gráficas. Por tanto, he hecho un estudio más profundo contabilizando cuántas slices “buenas” (accuracy mayor del 75 %) hay para cada corte. Los resultados son los siguientes:

	Accuracy medio	Número de buenas	Slices buenas	Accuracy	Accuracy Máximo
Axial	0.5554140	0	-	-	<0.75
Coronal	0.5974603	4	21,79, 140,145	0.767, 0.767, 0.833, 0.767	0.833
Sagittal	0.5866421	3	36,39,44	0.799, 0.756, 0.799	0.799

Tabla 4.1: Resultados del experimento 1

Identificamos que el plano Coronal (Y) es el que tiene mayor accuracy medio y mayor número de slices con accuracy alto, por lo que lo utilizamos en el resto de experimentos.

4.2. Experimento 2

En este experimento, utilizamos 165 (65 del grupo de control y 100 enfermos de PD) imágenes MRI para estudiar cómo se comporta el rendimiento de clasificación utilizando sólo la información de la materia gris, de la materia blanca o del cerebro completo. Presentamos varias gráficas comparativas para cada tipo de imagen en cada corte del plano Coronal.

4.2.1. Material y metodología

Las imágenes de los sujetos fueron, en primer lugar, segmentadas a través de la rutina de segmentación implementada en SPM12 ([18]). Tras ello, normalizamos y cambiamos el tamaño de la imagen a los bordes [-78 -112 -60; 78 76 85]. Extraemos características con la transformada discreta Wavelets 2D usando la transformada Bior3.3 hasta nivel 2. Tras ello, seleccionamos las características que explican el 95 % de la variabilidad en el conjunto de datos con PCA. Por último, el resultado de PCA fue ejecutado en SVM usando validación cruzada de 10 particiones y un conjunto de entrenamiento del 70 % de los datos.

Computamos los coeficientes wavelets 2D para cada slice y para cada paciente y llevamos a cabo la clasificación. Cada vez que entrenamos un SVM se realiza una permutación aleatoria de los pacientes para evitar el sesgo, de forma que el modelo no sobre ajusta, y realizamos 10 clasificaciones cada vez, quedándonos con la media de los resultados por cada slice.

Llevamos a cabo este proceso para C1 (materia gris), C2 (materia blanca) y C1+C2 de las imágenes.

4.2.2. Resultados

El objetivo de este experimento era comprobar qué tipo de información es más interesante para obtener el máximo accuracy en la clasificación. Además, descartar una parte de la imagen hace que el tiempo de cómputo general se reduzca mucho.

Presentamos los siguientes gráficos: En primer lugar, una imagen que muestra los resultados del accuracy para C1 y C2 superpuestos. A continuación, la diferencia de accuracy C2-C1 en cada slice (para ver la tendencia). Por último, presentamos el número de componentes principales para cada tipo de información.

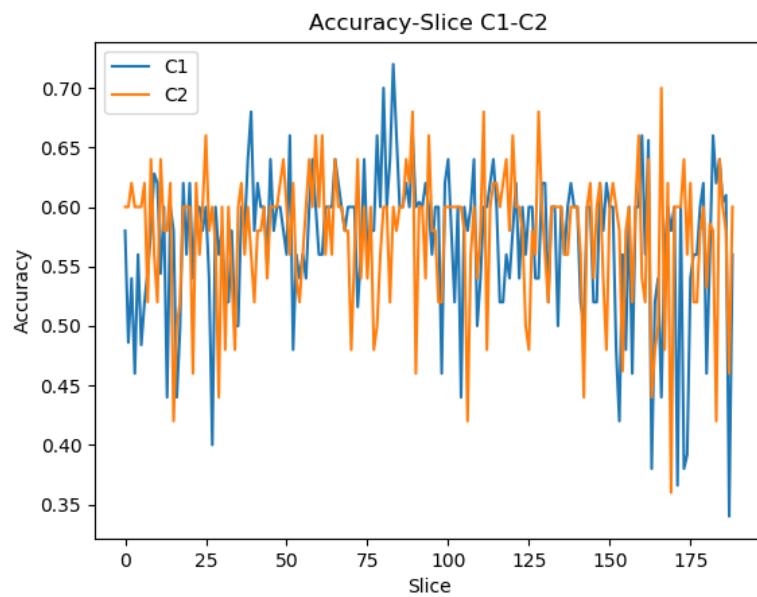


Figura 4.4: Comparativa de accuracy entre C1 y C2

No vemos claramente qué tipo de información es más interesante. Veamos la diferencia:

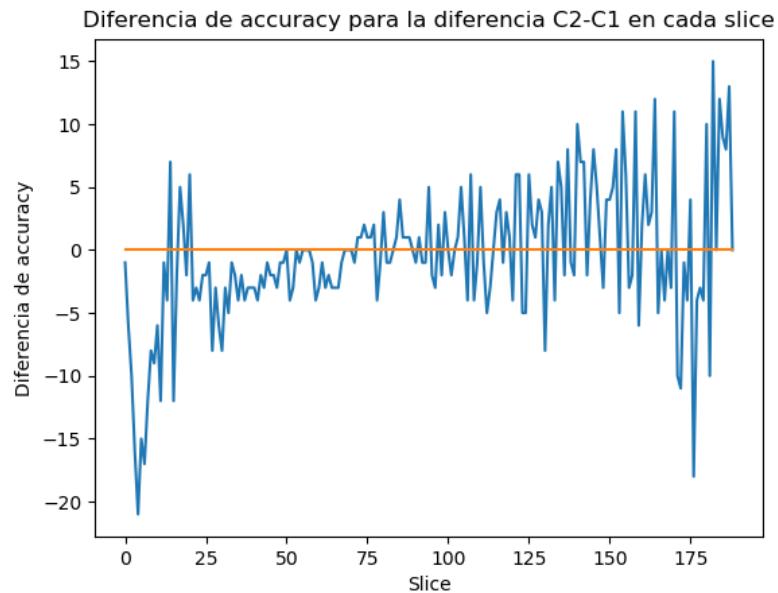


Figura 4.5: Diferencia de accuracy entre C2 y C1 para cada slice

La línea naranja representa las slices donde el accuracy entre C2 y C1 es igual. Lo que

queda por encima de la recta indica que C2 tiene mejor accuracy y viceversa. Veamos ahora el número de componentes principales para cada tipo de información.

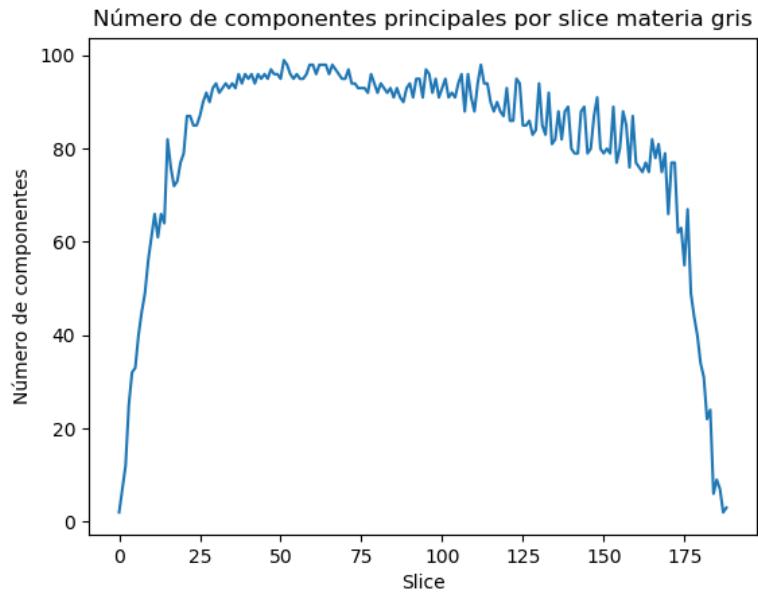


Figura 4.6: Número de componentes principales por slice para C1

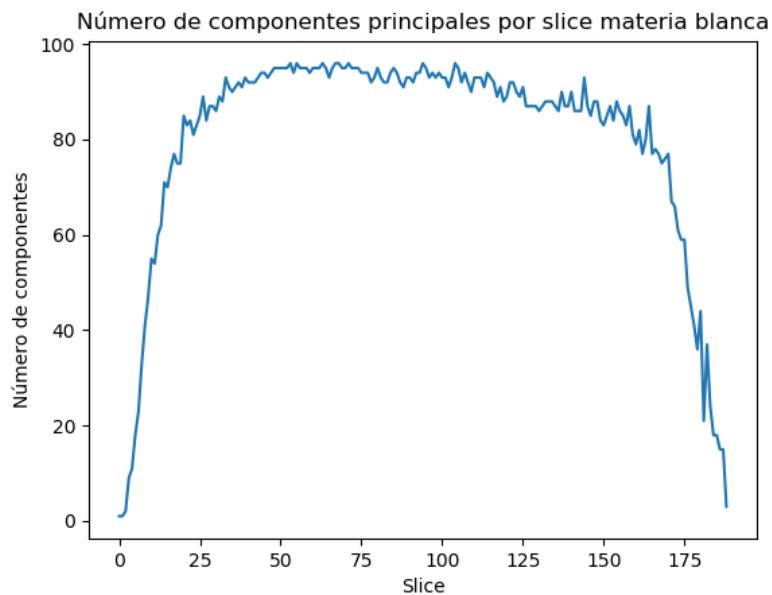


Figura 4.7: Número de componentes principales por slice para C2

4.2.3. Conclusiones

La conclusión que extraemos es que ambos tipos de información tienen un comportamiento similar en rendimiento de clasificación. De entre el corte 0 a 1 75 parece predominar la materia gris, y hasta el 189 la materia blanca obtiene ventaja. Se puede observar que la materia gris genera el máximo score entorno al corte 85. Además, desde el punto de vista estadístico, no se puede asegurar que una contenga más información relevante que la otra, por lo que optamos por seguir con toda la información del cerebro en los siguientes experimentos, a pesar de el incremento en el coste computacional que supone.

4.3. Experimento 3

En este experimento, utilizamos 242 (78 del grupo de control y 164 enfermos de PD) imágenes MRI para estudiar qué slices o cortes son las más determinantes a la hora de diagnosticar un paciente con Parkinson. A la luz del experimento 1, utilizamos plano Coronal como base y, por experimento 2, usamos la imagen completa del cerebro.

4.3.1. Material y metodología

Hacemos una partición de las imágenes. Por un lado, los pacientes que entran en juego en la primera fase de stacking (pacientes_training) y, por otro, los que serán evaluados en la segunda fase (pacientes_test). Todas las imágenes de los sujetos fueron, en primer lugar, segmentadas a través de la rutina de segmentación implementada en SPM12 ([18]). Tras ello, normalizamos y cambiamos el tamaño de la imagen a los bordes [-78 -112 -60; 78 76 85]. Extraemos características con la transformada discreta Wavelets 2D usando la transformada Bior3.3 hasta nivel 2. A continuación, realizo una normalización estándar de los coeficientes y, después, seleccionamos las características que explican el 95 % de la variabilidad en el conjunto de datos con PCA. Por último, el resultado de PCA fue ejecutado en SVM usando validación cruzada de 10 particiones y un conjunto de entrenamiento del 70 % de los datos.

Computamos los coeficientes wavelets 2D para cada slice, recopilamos toda la información en una matriz, generamos un modelos con SVM y el conjunto de training y, a continuación, hacemos una predicción con el conjunto de test, concluyendo la primera fase. Por tanto, hasta este momento, tenemos un nuevo dataset formado por las predicciones (valores 0 o 1) de un conjunto de pacientes de test, en el que en cada fila tenemos un paciente y cada columna corresponde a la predicción que nos ha dado cada slice sobre si el sujeto está enfermo o no (189 columnas).

En la segunda fase, dividimos de nuevo ese conjunto de datos final en entrenamiento y test (para evitar sobre aprendizaje) y, sobre la parte de train, aplico una regresión logística con validación cruzada de 10 particiones, para así poder valorar qué columnas (slices) tienen mayor peso a la hora del diagnóstico. Elijo un modelo lineal para poder conseguir una interpretabilidad mayor. Tras evaluar el modelo, conseguimos un accuracy total de 0.7 y, además, extraemos del mismo los pesos de cada slice para así poder ver cuáles son más determinantes.

4.3.2. Resultados

A parte del 0.7 en accuracy general, que no es un resultado que dé demasiada confianza, extraemos los 189 coeficientes asociados a los cortes y los parámetros del modelo. Nos centramos en los pesos. Son los siguientes:

[0.00000000e+00	-2.72081866e-04	0.00000000e+00	0.00000000e+00
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
1.41360676e-05	0.00000000e+00	0.00000000e+00	-4.16556476e-04	
-1.35084727e-05	2.71858398e-07	0.00000000e+00	-1.68421946e-05	
0.00000000e+00	0.00000000e+00	1.55596651e-04	0.00000000e+00	
-6.33401954e-06	1.38191410e-05	3.47858563e-05	-7.53403593e-05	
0.00000000e+00	0.00000000e+00	2.51681329e-04	-1.89061115e-04	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	5.70611828e-07	0.00000000e+00	0.00000000e+00	
5.70611828e-07	5.70611828e-07	0.00000000e+00	1.44827794e-04	
5.70611828e-07	0.00000000e+00	0.00000000e+00	0.00000000e+00	
5.70611828e-07	0.00000000e+00	-1.41225255e-04	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	5.70611828e-07	0.00000000e+00	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	5.70611828e-07	
5.70611828e-07	5.70611828e-07	0.00000000e+00	0.00000000e+00	
5.70611828e-07	5.70611828e-07	0.00000000e+00	5.70611828e-07	
0.00000000e+00	0.00000000e+00	7.28114621e-05	0.00000000e+00	
5.70611828e-07	0.00000000e+00	-2.20098109e-04	5.70611828e-07	
5.70611828e-07	5.70611828e-07	-1.20173868e-04	5.70611828e-07	
5.70611828e-07	0.00000000e+00	0.00000000e+00	5.70611828e-07	
0.00000000e+00	5.70611828e-07	0.00000000e+00	5.70611828e-07	
0.00000000e+00	5.70611828e-07	5.70611828e-07	1.10790011e-04	
5.70611828e-07	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	5.70611828e-07	0.00000000e+00	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
-3.10189170e-05	0.00000000e+00	5.86691900e-05	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
-4.09689633e-05	-4.42844578e-05	-6.52934031e-05	0.00000000e+00	
-3.73916070e-05	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	-3.73492089e-05	0.00000000e+00	0.00000000e+00	
0.00000000e+00	7.58739705e-05	0.00000000e+00	0.00000000e+00	
6.89822920e-05	-1.40862768e-04	-2.71207202e-05	4.84346823e-05	
0.00000000e+00	2.43046660e-05	0.00000000e+00	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	0.00000000e+00	-2.02858942e-04	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	9.65823358e-05	0.00000000e+00	6.89658532e-05	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	-4.44282380e-05	0.00000000e+00	-1.13402775e-04	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00]				

Figura 4.8: Pesos asociados a cada slice en la regresión logística para el diagnóstico.
Experimento 3

Como se puede observar, la mayoría de ellos son cero. Esto quiere decir que no aportan ninguna información en el diagnóstico (como cabría esperar, dado que, como se explicó en la sección 1, la enfermedad aparece en el mesencéfalo, en unas zonas muy concretas, por lo que un corte cercano a la parte superior de la cabeza no debería darnos información, por ejemplo), unos son positivos y otros negativos. Encontramos en la literatura el argumento de que aquellos coeficientes positivos más grandes son los que más información aportan para clasificar en la clase positiva, en nuestro caso, a los sujetos en el grupo de control. Por el contrario, aquellos coeficientes negativos que tienen mayor valor absoluto dan información al diagnóstico de enfermos. Por tanto, podemos darle una doble información al médico. En principio nos centramos en los coeficientes que dan información sobre el enfermo. Si tomamos los 10 mejores cortes, obtenemos:

$$[11, 1, 78, 162, 27, 50, 149, 82, 183, 23]$$

que se traduce, en una imagen cualquiera de un cerebro con plano Axial y Sagittal, en

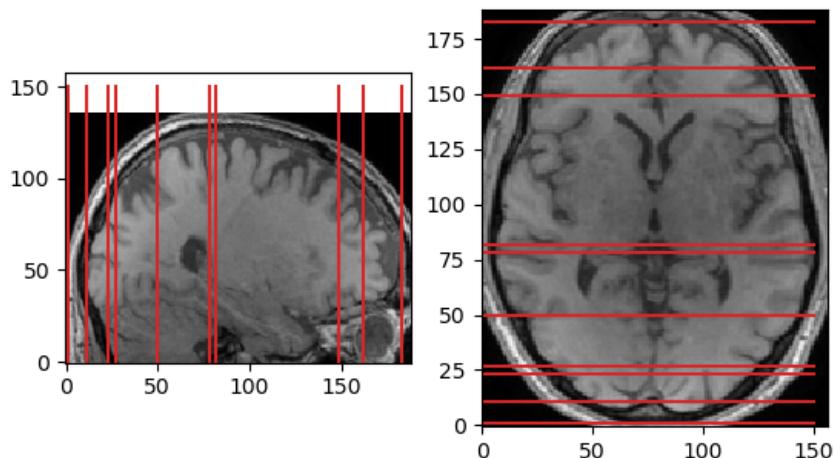


Figura 4.9: Cortes más determinantes para Parkinson. Experimento 3

En el caso de los más determinantes para el grupo de control, encontramos

$$[26, 18, 43, 95, 173, 145, 74, 148, 175, 118]$$

obteniendo la imagen

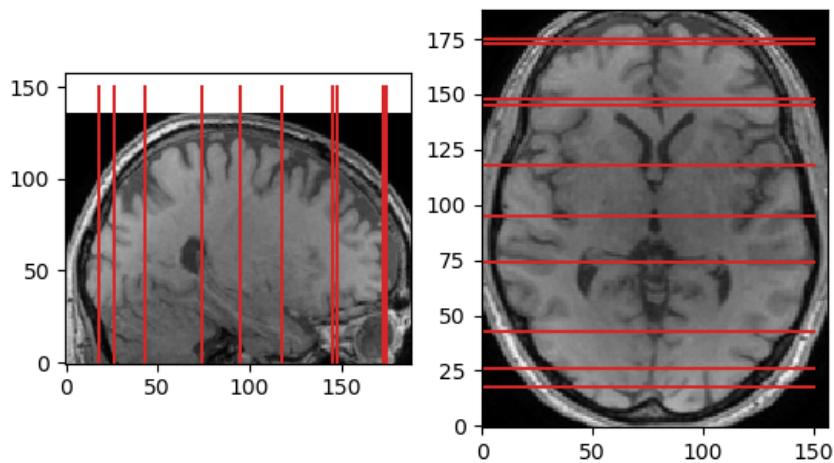


Figura 4.10: Cortes más determinantes para Control. Experimento 3

Como no hay estudios médicos hasta la fecha sobre imágenes MRI y Parkinson, no podemos comprobar si los resultados se ajustan a certezas médicas. En [15] se utilizan algoritmos genéticos en la optimización y se obtienen slices distintas.

Clasificación

Hacemos una nueva clasificación incluyendo únicamente los slices vinculantes encontrados, utilizando el mismo preprocesamiento que antes y con un GridSearch de parámetros para SVM con validación cruzada de 10 particiones y 30 % de datos para el test (se puede encontrar en *experimento3-bs.py*). Los resultados son los siguientes:

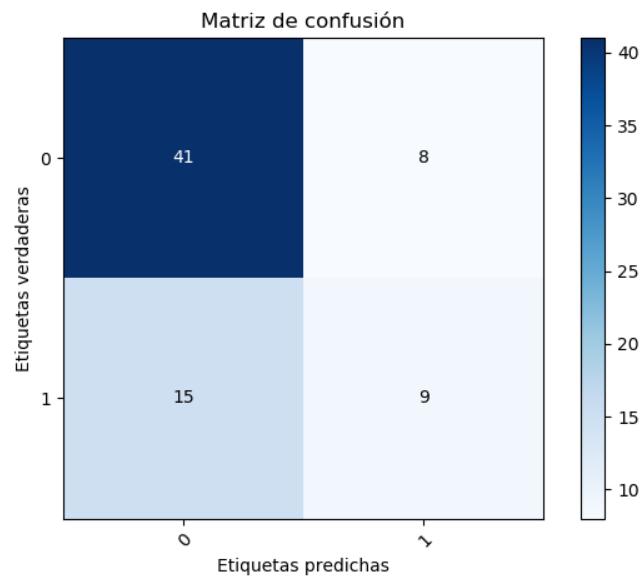


Figura 4.11: Matriz de confusión para la clasificación con los mejores cortes

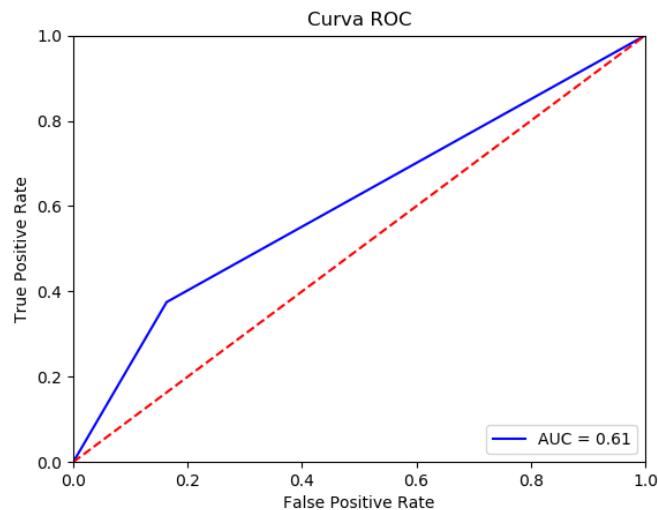


Figura 4.12: Curva ROC para la clasificación con los mejores cortes

Resultados para la predicción de Parkinson:

- Precision: 0.73
- Recall: 0.84
- F1-Score: 0.78

4.4. Experimento 4

En este experimento, utilizamos 242 (78 del grupo de control y 164 enfermos de PD) imágenes MRI para estudiar qué slices o cortes son las más determinantes a la hora de diagnosticar un paciente con Parkinson. A la luz del experimento 1, utilizamos plano Coronal como base y, por experimento 2, usamos la imagen completa del cerebro.

4.4.1. Material y metodología

Volvemos a realizar la misma partición que en el experimento 3. Sin embargo, en esta ocasión no hacemos preprocesamiento alguno, buscando aprovechar las bondades de las redes neuronales profundas. La primera parte de los pacientes, para la fase uno del stacking, será utilizada para entrenar la red. Se genera un modelo por cada slice y se van almacenando las predicciones generadas con los elementos de la segunda fase. Dichas predicciones son evaluadas por la regresión logística con validación cruzada de 10 particiones para valorar los cortes más importantes según sus pesos, tal y como hicimos en el experimento anterior.

Antes de presentar los resultados, explico con detalle la topología de la red. Partimos de un modelo con dos capaz convolutivas que trabajan con la imagen completa ([34]):

Layer (type)	Output Shape	Param #
conv2d_19 (Conv2D)	(None, 155, 134, 64)	640
conv2d_20 (Conv2D)	(None, 153, 132, 32)	18464
flatten_10 (Flatten)	(None, 646272)	0
dense_10 (Dense)	(None, 2)	1292546
<hr/>		
Total params:	1,311,650	
Trainable params:	1,311,650	
Non-trainable params:	0	

Figura 4.13: Configuración inicial de la red

Dado el número reducido de imágenes del que disponemos, parece excesivo tener más de 1.3 millones de parámetros ajustables por cada slice. A partir de aquí, añado capas de submuestreo (pooling) y una capa adicional densa. Además, reduzco el tamaño de la imagen con una tercera capa convolutiva (seguida de su capa de pooling):

Layer (type)	Output Shape	Param #
conv2d_49 (Conv2D)	(None, 155, 134, 32)	320
max_pooling2d_49 (MaxPooling)	(None, 77, 67, 32)	0
conv2d_50 (Conv2D)	(None, 75, 65, 32)	9248
max_pooling2d_50 (MaxPooling)	(None, 37, 32, 32)	0
conv2d_51 (Conv2D)	(None, 35, 30, 32)	9248
max_pooling2d_51 (MaxPooling)	(None, 17, 15, 32)	0
flatten_17 (Flatten)	(None, 8160)	0
dense_33 (Dense)	(None, 32)	261152
dense_34 (Dense)	(None, 2)	66
=====		
Total params:	280,034	
Trainable params:	280,034	
Non-trainable params:	0	

Figura 4.14: Diseño de la red, parte 1

Aún así, siguen siendo demasiados parámetros para el tamaño del conjunto de datos que tenemos, por lo que reduzco aún más el tamaño de la red reduciendo su número de canales.

Layer (type)	Output Shape	Param #
conv2d_118 (Conv2D)	(None, 155, 134, 32)	320
max_pooling2d_118 (MaxPooling2D)	(None, 77, 67, 32)	0
conv2d_119 (Conv2D)	(None, 75, 65, 16)	4624
max_pooling2d_119 (MaxPooling2D)	(None, 37, 32, 16)	0
conv2d_120 (Conv2D)	(None, 35, 30, 16)	2320
max_pooling2d_120 (MaxPooling2D)	(None, 17, 15, 16)	0
flatten_40 (Flatten)	(None, 4080)	0
dense_79 (Dense)	(None, 16)	65296
dense_80 (Dense)	(None, 2)	34
Total params:	72,594	
Trainable params:	72,594	
Non-trainable params:	0	

Figura 4.15: Diseño de la red, parte 2

Fijo el número de épocas a 100. Por si la red no aprendiera lo suficiente en las primeras épocas, introduzco la posibilidad de detener el entrenamiento usando “early stopping”, de forma que podemos ahorrar gran parte del tiempo de cálculo, ya que resultaría innecesario.

4.4.2. Resultados

Obtenemos un 0.68 de accuracy en la validación cruzada. Estos son los coeficientes obtenidos para los 189 cortes:

[4.62702964e-07	4.62702964e-07	4.62702964e-07	4.62702964e-07
4.62702964e-07	4.62702964e-07	0.00000000e+00	8.96210898e-05	
0.00000000e+00	1.41619113e-04	4.62702964e-07	0.00000000e+00	
-4.82171493e-05	4.62702964e-07	0.00000000e+00	0.00000000e+00	
4.62702964e-07	4.62702964e-07	0.00000000e+00	-3.10121009e-05	
4.62702964e-07	-3.10265522e-05	0.00000000e+00	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	1.03539524e-04	
0.00000000e+00	1.20881919e-04	-7.92139983e-05	4.62702964e-07	
0.00000000e+00	2.06877479e-04	4.62702964e-07	0.00000000e+00	
4.62702964e-07	0.00000000e+00	4.62702964e-07	4.62702964e-07	
0.00000000e+00	3.81084813e-05	0.00000000e+00	4.62702964e-07	
0.00000000e+00	0.00000000e+00	4.62702964e-07	-3.10070969e-05	
4.62702964e-07	0.00000000e+00	4.62702964e-07	0.00000000e+00	
0.00000000e+00	-3.10122477e-05	-9.30472193e-05	4.62702964e-07	
4.62702964e-07	4.62702964e-07	-3.89089151e-04	4.62702964e-07	
4.62702964e-07	0.00000000e+00	4.62702964e-07	8.63191885e-05	
-1.75371627e-04	0.00000000e+00	4.62702964e-07	0.00000000e+00	
0.00000000e+00	0.00000000e+00	4.62702964e-07	4.62702964e-07	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	-2.44331977e-04	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	4.62702964e-07	4.62702964e-07	4.62702964e-07	
4.62702964e-07	4.62702964e-07	0.00000000e+00	4.62702964e-07	
0.00000000e+00	0.00000000e+00	4.62702964e-07	0.00000000e+00	
0.00000000e+00	9.31965048e-05	4.62702964e-07	4.62702964e-07	
4.62702964e-07	0.00000000e+00	0.00000000e+00	0.00000000e+00	
-3.10239287e-05	-1.40902366e-04	3.81168961e-05	0.00000000e+00	
0.00000000e+00	0.00000000e+00	1.41657500e-04	0.00000000e+00	
0.00000000e+00	0.00000000e+00	4.62702964e-07	0.00000000e+00	
4.62702964e-07	-3.41107479e-05	-3.10351374e-05	0.00000000e+00	
0.00000000e+00	0.00000000e+00	6.89571308e-05	0.00000000e+00	
-1.33808770e-05	4.62702964e-07	2.91937543e-07	6.95602698e-06	
4.62702964e-07	4.62702964e-07	4.62702964e-07	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	0.00000000e+00	-1.72099514e-05	4.62702964e-07	
0.00000000e+00	0.00000000e+00	4.62702964e-07	0.00000000e+00	
0.00000000e+00	-3.02920452e-04	4.62702964e-07	4.62702964e-07	
4.62702964e-07	0.00000000e+00	4.62702964e-07	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	4.62702964e-07	4.62702964e-07	4.62702964e-07	
0.00000000e+00	4.62702964e-07	0.00000000e+00	4.62702964e-07	
0.00000000e+00	0.00000000e+00	0.00000000e+00	4.62702964e-07	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	
4.62702964e-07	4.62702964e-07	0.00000000e+00	0.00000000e+00	
0.00000000e+00	0.00000000e+00	4.62702964e-07	0.00000000e+00	
-1.03278986e-04	0.00000000e+00	0.00000000e+00	4.62702964e-07	
4.62702964e-07	0.00000000e+00	0.00000000e+00	3.79544076e-05	
0.00000000e+00]				

Figura 4.16: Pesos asociados a cada slice en la regresión logística para el diagnóstico.
Experimento 4

De nuevo, estudiamos los 10 mejores cortes para el diagnóstico del Parkinson y para el grupo de Control:

[58, 145, 79, 64, 105, 180, 54, 30, 12, 117]

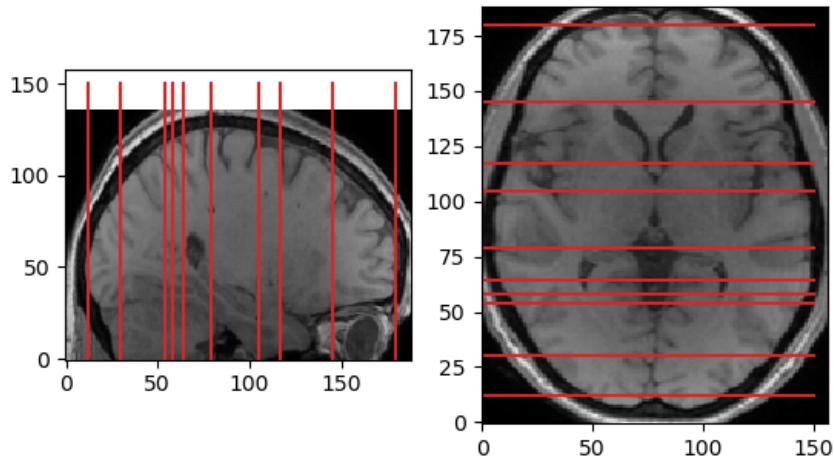


Figura 4.17: Cortes más determinantes para Parkinson. Experimento 4

Para el grupo de Control,

[33, 110, 9, 29, 27, 97, 7, 63, 122, 106]

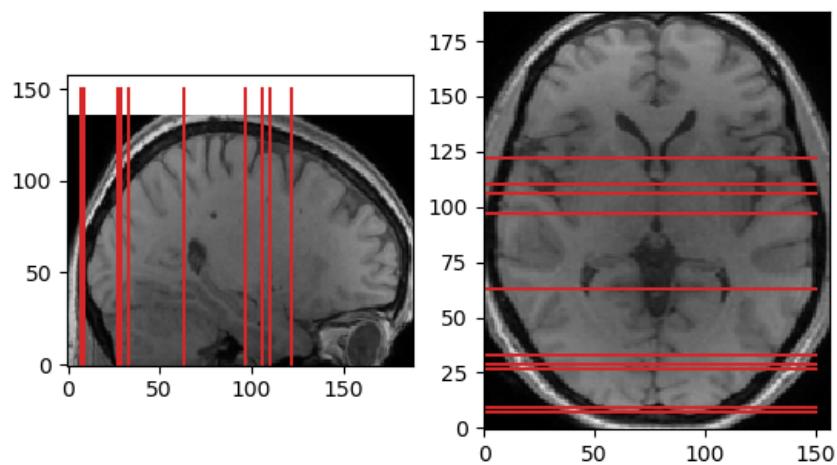


Figura 4.18: Cortes más determinantes para Control. Experimento 4

CLASIFICACIÓN

Realizamos una clasificación con el mismo propósito que en el experimento 3 (*experimento4_bs.py*). Sin embargo, los resultados no han sido tan buenos. Basta ver la matriz de confusión para advertir que la red no ha conseguido aprender las características de la enfermedad con claridad:

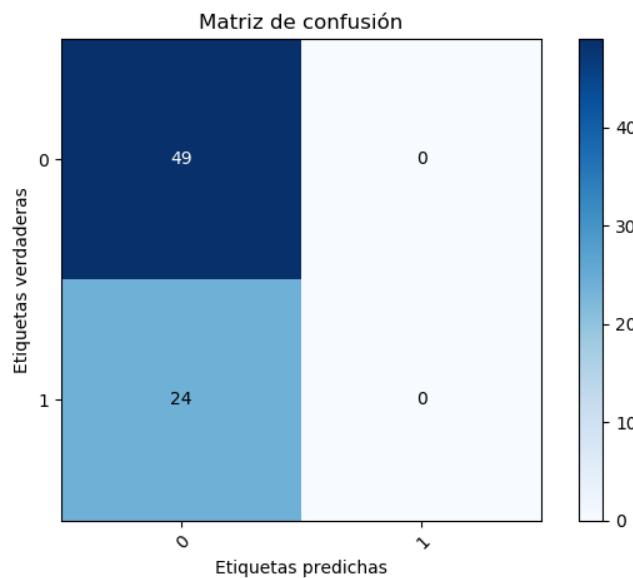


Figura 4.19: Matriz de confusión para la clasificación con los mejores cortes

Como se puede ver, ha etiquetado a todos los elementos del test como si fueran enfermos, lo cuál es grave, ya que tendríamos gran cantidad de falsos positivos. A la luz de los resultados, no parece que deep learning pueda tener un buen rendimiento en un problema con tan pocos datos.

5. Desarrollo del proyecto software

5.1. Análisis y diseño

5.1.1. Subsistemas

El sistema de información desarrollado se divide en 5 subsistemas, que son: Subsistema de normalización, Subsistema experimento 1, Subsistema experimento 2, Subsistema experimento 3 y Subsistema experimento 4. El primero de ellos es auxiliar y se encarga de normalizar, segmentar y generar las distintos tipos de fuentes de datos (C1, C2 y cerebro completo) para el resto de subsistemas. Los siguientes llevan a cabo cada uno de los experimentos descritos y justificados en la sección anterior. A continuación se presentan los requisitos de datos y funcionales del sistema completo.

5.1.2. Requisitos de datos

R.D. 1: Imágenes en bruto de MRI 3D en formato .nii (originalmente DICOM y transformado después).

R.D. 2: Imágenes segmentadas y normalizadas en formato .nii con tres tipos de información: materia blanca, gris y cerebro completo.

R.D. 3: Imágenes preprocesadas C1.

R.D. 4: Imágenes preprocesadas C2.

R.D. 5: Imágenes preprocesadas cerebro completo.

R.D. 6: Características vía coeficientes de transformada discreta Wavelet 2D.

R.D. 7: Características vía PCA.

R.D. 8: Resultados de los experimentos: Gráficas, datos, valores de rendimiento, medidas.

R.D. 9: Predicciones sobre el conjunto de test en stacking (2^a fase).

5.1.3. Requisitos funcionales

R.F. 1: Normalizar y segmentar las imágenes.

- Datos de entrada: R.D.1
- Datos de salida: R.D.2

R.F. 2: Extracción de características con wavelets 2D.

- Datos de entrada: R.D.2
- Datos de salida: R.D.6

R.F. 3 Selección de características con PCA.

- Datos de entrada: R.D.6

- Datos de salida: R.D.7

R.F. 4 Obtención del mejor plano.

- Datos de entrada: R.D.7
- Datos de salida: R.D.8

R.F. 5 Obtención de la mejor fuente de información.

- Datos de entrada: R.D.7 (vía R.D. 3, 4 y 5)
- Datos de salida: R.D.8

R.F. 6 Optimización de slices.

- Datos de entrada: R.D.9
- Datos de salida: R.D.8

5.1.4. Diagramas

ESQUEMA DE CAJA NEGRA

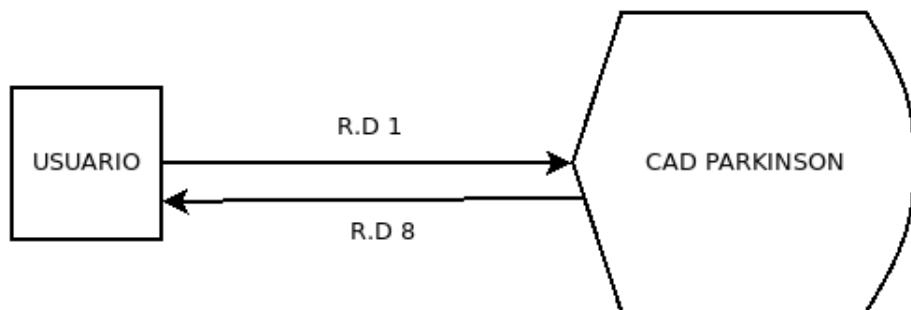


Figura 5.1: Esquema de caja negra

ESQUEMA F-ARMAZÓN

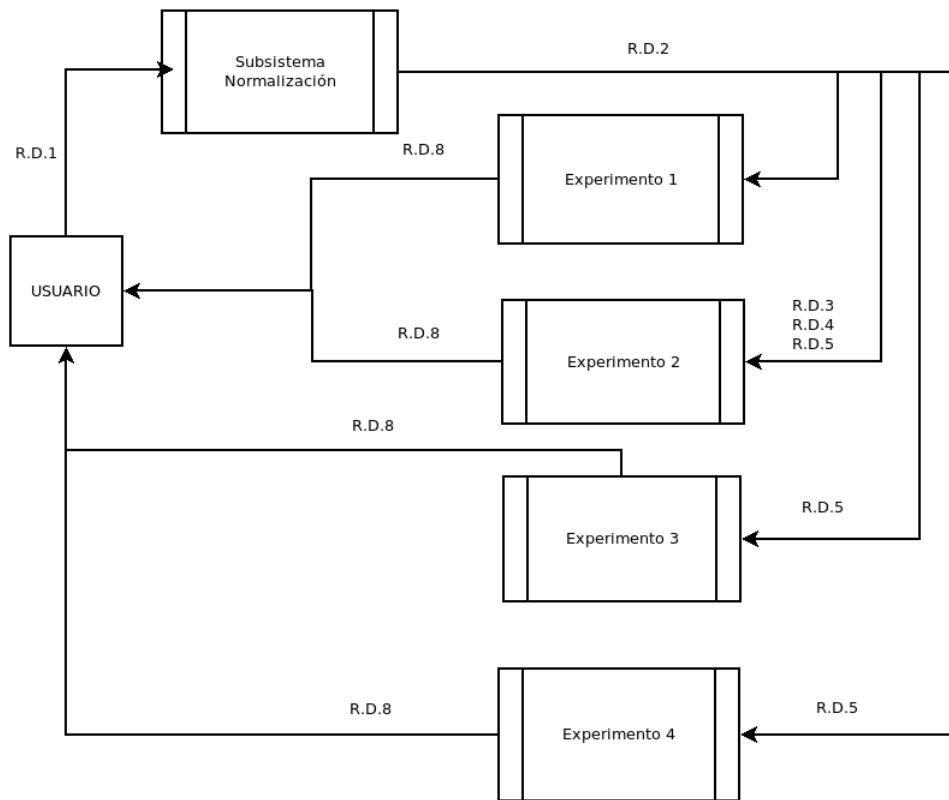


Figura 5.2: Esquema F-Armazón de los subsistemas

FLUJO DE DATOS EXPERIMENTO 1

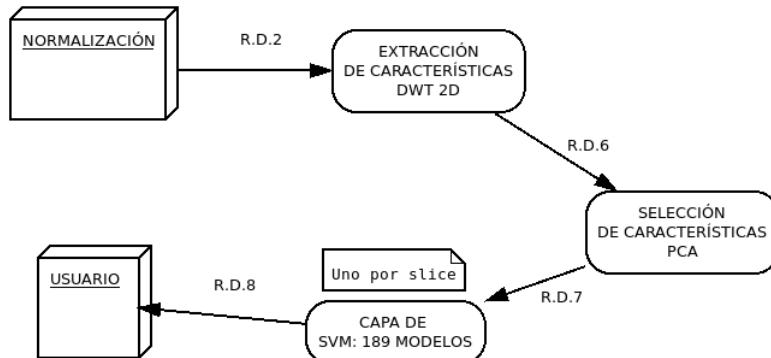


Figura 5.3: Esquema de flujo de datos para Experimento 1

FLUJO DE DATOS EXPERIMENTO 2

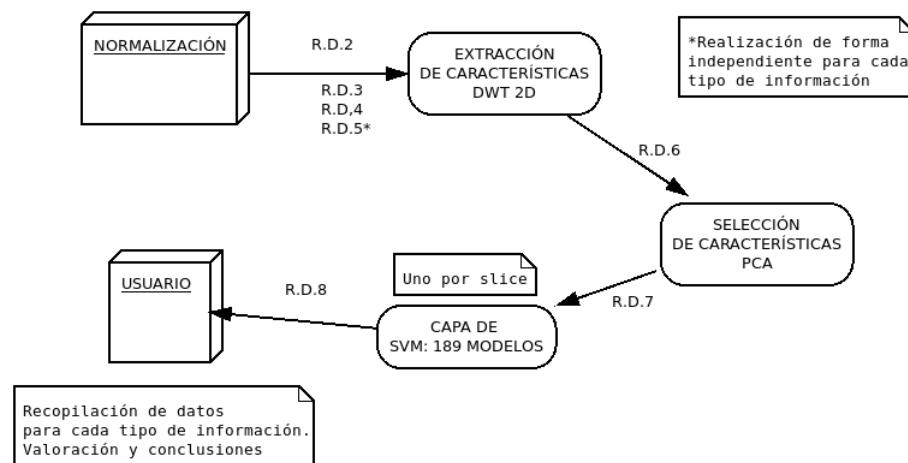


Figura 5.4: Esquema de flujo de datos para Experimento 2

FLUJO DE DATOS EXPERIMENTO 3

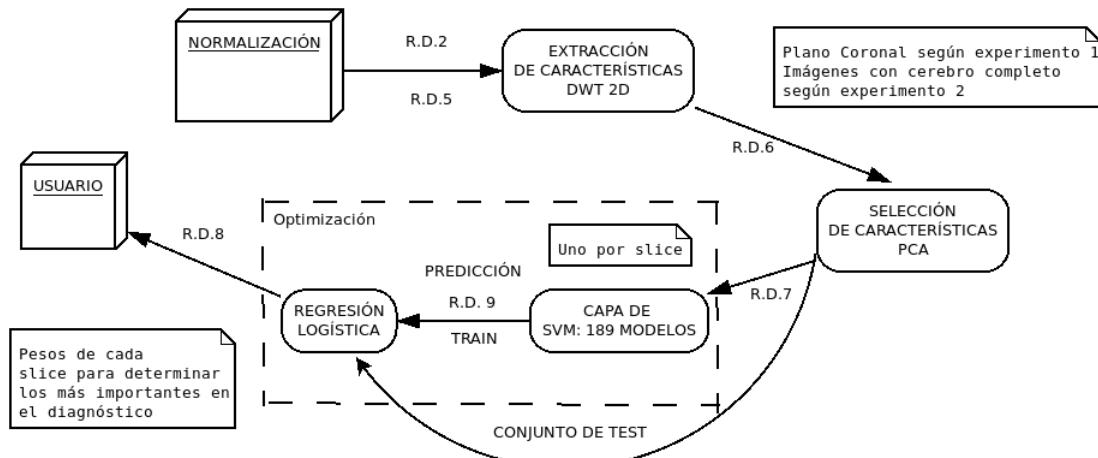


Figura 5.5: Esquema de flujo de datos para Experimento 3

FLUJO DE DATOS EXPERIMENTO 4

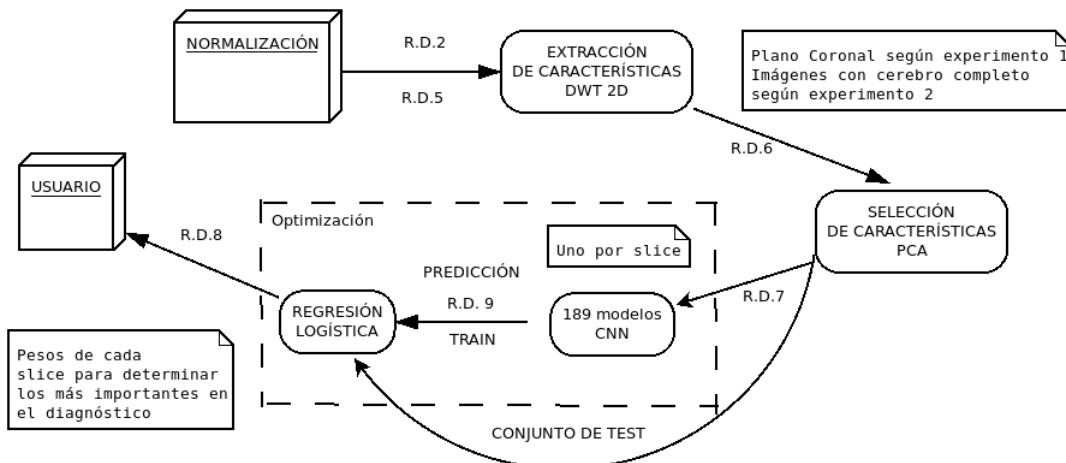


Figura 5.6: Esquema de flujo de datos para Experimento 4

5.2. Implementación

Este proyecto software tiene licencia GNU General Public License v3.0, y todo su contenido se encuentra en mi repositorio <https://github.com/luisbalru/TFG>. La totalidad de su desarrollo se ha hecho en Python a excepción del subsistema de normalización en el que empleo Matlab, ya que no encontré las herramientas necesarias ni las bibliotecas en Python. Como bibliotecas especiales, cabe destacar SPM12 ([18]) para la normalización y segmentación, Nibabel para lectura de neuroimágenes, SKlearn en general para todo el tratamiento, pywt para la extracción de características con DWT 2D, TensorFlow y Keras para la parte de Deep Learning. Para su utilización, basta con ejecutar los distintos ficheros “experimento_num-exp.py” con la previa instalación de las dependencias. No necesita ningún argumento de entrada y se ejecuta sobre las imágenes situadas en las carpetas *Datos/PD* o *Datos/Control* para pacientes enfermos y sanos, respectivamente. Los resultados de los experimentos se guardan en ficheros de texto en el mismo directorio donde se encuentran los fuentes y son interpretados con los “res-exp-num-exp.py”, apareciendo por pantalla los gráficos y/o resultados numéricos de precisión.

6. Conclusiones y proyectos futuros

6.1. Conclusiones

El sistema de ayuda al diagnóstico del Parkinson desarrollado ha constado de cuatro experimentos destinados al incremento paulatino de nuestro conocimiento sobre la enfermedad y el funcionamiento del cerebro. El primero de ellos estaba destinado a, de entre los tres cortes posibles que podemos hacer para obtener una imagen de dos dimensiones (axial, coronal y sagittal), cuál es más representativa. Nuestro resultado fue contundente: el plano coronal es el más informativo de todos ellos para esta demencia. El siguiente, que tenía propósito de ahorrar coste computacional, consistía en decidir si podíamos prescindir de una parte de la imagen completa del cerebro, quedándonos con la materia gris (C1) o la materia blanca (C2). Los resultados no fueron concluyentes, por lo que optamos por tomar siempre la imagen completa, a fin de garantizar la información más rica y los resultados más interesantes posibles. El tercer experimento tenía como objetivo primordial darle respuesta a la pregunta de qué corte, de los 189 que genera el plano coronal, contiene mayor información de interés en el diagnóstico de la enfermedad del Parkinson. Como es obvio, esta es nuestra pregunta central y el objeto de todo nuestro estudio. En la sección de experimentos se detallan los resultados, pero se puede decir que nuestro sistema es capaz de seleccionar las 10 slices más importantes y detecta a pacientes enfermos con una precisión del 73 % y un recall del 84 %. El experimento 4 estaba dedicado a la optimización vía deep learning de nuestro sistema. A pesar de que los resultados no han sido buenos, el diseño de la red ha sido enriquecedor y construye una nueva vía para el desarrollo de la biomedicina y el diagnóstico de enfermedades neurodegenerativas.

6.2. Proyectos futuros

Existen trabajos pioneros ([14]) que tratan de diagnosticar enfermedades neurodegenerativas trabajando con imágenes completas en 3D y no por cortes. La extensión de este trabajo de software libre a ese terreno sería un proyecto futuro muy interesante y viable, haciendo así herramientas abiertas para toda la comunidad. Además, deberíamos plantearnos utilizar otro tipo de imágenes, ya que expertos en imagen y radiología me advirtieron que MRI no era la mejor fuente de información para el diagnóstico de esta enfermedad. El traslado de estas técnicas a imágenes más potentes puede darnos un salto de calidad y verosimilitud que por fin consiga convencer a los médicos de la importancia de estas herramientas. Nunca fue nuestro propósito hacer de médicos ni acallar su criterio, si no darles una alternativa en casos de duda.

Desde un punto de vista más técnico, los resultados del experimento 4 confirman que deep learning no responde bien a problemas en los que el número de datos es reducido. Optamos por no utilizar *data augmentation* para no falsear los resultados ni aumentar las estadísticas. Por otra parte, la generación de 189 modelos de deep learning podría mejorarse creando una única red más ambiciosa, en la que gran mayoría de las capas (sobre todo las primeras) fueran comunes para todas las slices, ya que no hay grandes

diferencias al principio. Sólo se dividirían en detalles muy concretos, para que así la red consiguiera aprender con más esmero.

7. Código

7.1. Normalización

```
function [] = ppmi_normal(file)
addpath(genpath('NIfTI 20140122'));
addpath(genpath('spm12'));
% This function just normalizes the input file to MNI space, in a bounding
% box of 157x189x136 voxels, segments image into gray and white matter,
% and store the results in w*.nii wc1*.nii and wc2*.nii
tic
%Create the file paths to GM and WM
[path,name,ext] = fileparts(file);

out = strcat(path,'/out');
c1 = fullfile(path,[strcat('c1',name) ext]);
c2 = fullfile(path,[strcat('c2',name) ext]);
% Path to mats:
filename = file(1:end-4);
toc

tic
%Image segmentation
start = strcat(file);
disp(start)
results = spm_preproc(start);
toc

tic
% Generation of the normalization parameters
[po, pin] = spm_prep2sn(results);
% Inverse parameters storage
VG = pin.VG;
VF = pin.VF;
Tr = pin.Tr;
Affine = pin.Affine;

flags = pin.flags;
save(fullfile(strcat(filename,'_seg_inv_sn.mat')),
      '-V6','VG','VF','Tr','Affine','flags');
toc
% Forward parameters storage
VG = po.VG;
VF = po.VF;
Tr = po.Tr;
Affine = po.Affine;
flags = po.flags;
transformationParameters = strcat(filename,'_seg_sn.mat');
```

```

fnam = fullfile(transformationParameters);
save(fnam, '-V6', 'VG', 'VF', 'Tr', 'Affine', 'flags');

%Segmentation with output parameters
spm_preproc_write(po);
%Setup of the spatial normalization parameters
defaults.normalise.write.preserve = reshape(double(0), [1,1]);
defaults.normalise.write.bb = reshape(double([-78 78 -112 76 -50 85]),[2,3]);
defaults.normalise.write.vox = reshape(double([1 1 1]), [1,3]);
defaults.normalise.write.interp = reshape(double([1]), [1,1]);
defaults.normalise.write.wrap = reshape(double([0 0 0]), [1,3]);
tic

% Image normalization
% Normal image
spm_write_sn(file, transformationParameters,defaults.normalise.write);
% Grey matter image
spm_write_sn(c1,transformationParameters, defaults.normalise.write);
% White matter image
spm_write_sn(c2, transformationParameters, defaults.normalise.write);
toc

end

```

7.2. Sujetos

```

# Clase para estructurar a los sujetos en memoria
# AUTOR: LUIS BALDERAS RUIZ
# TRABAJO FINAL DE GRADO

import nibabel as nib
import pywt

class Sujeto:
    def __init__(self, image_name,group):
        self.nombre_image = image_name
        self.grupo = group
        img = nib.nifti1.load(image_name)
        self.img_data = img.get_fdata()
        self.shape_image = self.img_data.shape

    def get_label(self):
        return self.grupo

    def get_shape(self,plano):
        return self.shape_image[plano]

```

```

def get_slice(self,plano,slice):
    if plano == 0:
        return self.img_data[slice,:,:]
    elif plano == 1:
        return self.img_data[:,slice,:]
    elif plano == 2:
        return self.img_data[:, :, slice]
    else:
        print("Dio un plano que no existe")

def get_wave2D(self,slice,transform,lev):
    cA2, (cH2, cV2, cD2), (cH1, cV1, cD1) =
        pywt.wavedec2(slice,transform,level=lev)
    coeffs = []
    for i in range(len(cA2)):
        for j in range(len(cA2[0])):
            coeffs.append(cA2[i][j])

    return coeffs

```

7.3. Experimento 1

```

# SCRIPT PARA EL EXPERIMENTO 1
# Determinar que plano es mas determinante en la clasificacion
# Tratamiento por slices
# Author: Luis Balderas Ruiz

```

```

from subjects import Sujeto
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler

pacientes = []
imagenes_pd = os.listdir('./Datos/PD')
# Set the parameters by cross-validation

```

```

C_range = np.logspace(-2,10,13)
gamma_range = np.logspace(-9,3,13)
#C_range = np.logspace(-5, 15, 11)
#gamma_range = np.logspace(3, -15, 10)
param_grid = dict(gamma=gamma_range, C=C_range)
#param_grid = {'C': [1e3, 5e3, 1e4, 5e4, 1e5], 'gamma': [0.0001, 0.0005, 0.001,
    0.005, 0.01, 0.1], }

scores = ['recall']

print("LECTURA DE IMAGENES")

print("Leyendo pacientes enfermos")

for i in range(len(imagenes_pd)):
    imagenes_pd[i] = './Datos/PD/' + imagenes_pd[i]
    paciente = Sujeto(imagenes_pd[i],0)
    pacientes.append(paciente)

imagenes_control = os.listdir('./Datos/Control')

print("Leyendo pacientes control")

for i in range(len(imagenes_control)):
    imagenes_control[i] = './Datos/Control/' + imagenes_control[i]
    paciente = Sujeto(imagenes_control[i],1)
    pacientes.append(paciente)

print("X (Plano 0)")

X_shape = pacientes[0].get_shape(0)
accuracy_Xslices = []

for j in range(X_shape):
    print("Slice ", j)
    dataset = []
    target = []
    np.random.shuffle(pacientes)
    for i in range(len(pacientes)):
        slice = pacientes[i].get_slice(0,j)
        label = pacientes[i].get_label()
        row = pacientes[i].get_wave2D(slice,'bior3.3',2)
        dataset.append(row)
        target.append(label)

    scaler = StandardScaler()
    dataset = scaler.fit_transform(dataset)

```

```

X_train, X_test, y_train, y_test =
    train_test_split(dataset,target,test_size=0.3,random_state =
    77145416,stratify = target)
y_train = np.array(y_train)
pca = PCA(n_components = 0.95, svd_solver = 'full')
pca.fit(X_train)
X_train = pca.transform(X_train)
X_test = pca.transform(X_test)
clf = 0
for score in scores:
    clf = GridSearchCV(SVC(),param_grid=param_grid , cv=10,
                        scoring='%s_macro' % score)
    clf.fit(X_train, y_train)

mean_score_slice = 0
for k in range(0,10):
    indices = np.random.choice(len(X_train),size=len(X_train),replace=False)
    dataset_shuffle = X_train[indices]
    target_shuffle = y_train[indices]
    svm = SVC(C = clf.best_params_['C'],gamma = clf.best_params_['gamma'])
    svm.fit(dataset_shuffle,target_shuffle)
    y_true, y_pred = y_test, svm.predict(X_test)
    mean_score_slice = mean_score_slice + accuracy_score(y_true,y_pred)

mean_score_slice = mean_score_slice/10
accuracy_Xslices.append(mean_score_slice)

f = open('planoX-d.txt','a')
f.write('X (Plano 0)')
f.write(str(np.argmax(accuracy_Xslices)))
f.write(str(accuracy_Xslices[np.argmax(accuracy_Xslices)]))
f.close()
accuracy_Xslices = np.array(accuracy_Xslices)
accuracy_Xslices.tofile('accuracy_Xslices-d.out',sep=",")

print("Y (Plano 1)")

Y_shape = pacientes[0].get_shape(1)
accuracy_Yslices = []

for j in range(Y_shape):
    print("Slice ", j)
    dataset = []
    target = []
    np.random.shuffle(pacientes)
    for i in range(len(pacientes)):
        slice = pacientes[i].get_slice(1,j)
        label = pacientes[i].get_label()

```

```

row = pacientes[i].get_wave2D(slice,'bior3.3',2)
dataset.append(row)
target.append(label)

scaler = StandardScaler()
dataset = scaler.fit_transform(dataset)
X_train, X_test, y_train, y_test =
    train_test_split(dataset,target,test_size=0.3,random_state =
77145416,stratify = target)
y_train = np.array(y_train)
pca = PCA(n_components = 0.95, svd_solver = 'full')
pca.fit(X_train)
X_train = pca.transform(X_train)
X_test = pca.transform(X_test)
clf = SVC()
for score in scores:
    clf = GridSearchCV(SVC(), param_grid=param_grid, cv=10,
                       scoring='%s_macro' % score)
    clf.fit(X_train, y_train)

mean_score_slice = 0
for k in range(0,10):
    indices = np.random.choice(len(X_train),size=len(X_train),replace=False)
    dataset_shuffle = X_train[indices]
    target_shuffle = y_train[indices]
    svm = SVC(C = clf.best_params_['C'],gamma = clf.best_params_['gamma'])
    svm.fit(dataset_shuffle,target_shuffle)
    y_true, y_pred = y_test, svm.predict(X_test)
    mean_score_slice = mean_score_slice + accuracy_score(y_true,y_pred)

mean_score_slice = mean_score_slice/10
accuracy_Yslices.append(mean_score_slice)

g = open("planoY-d.txt","a")
g.write("Y (Plano 1)")
g.write(str(np.argmax(accuracy_Yslices)))
g.write(str(accuracy_Yslices[np.argmax(accuracy_Yslices)]))
accuracy_Yslices = np.array(accuracy_Yslices)
accuracy_Yslices.tofile('accuracy_Yslices-d.out',sep=",")

print("Z (Plano 2)")

Z_shape = pacientes[0].get_shape(2)
accuracy_Zslices = []

for j in range(Z_shape):
    print("Slice ", j)
    dataset = []
    target = []

```

```

np.random.shuffle(pacientes)
for i in range(len(pacientes)):
    slice = pacientes[i].get_slice(2,j)
    label = pacientes[i].get_label()
    row = pacientes[i].get_wave2D(slice,'bior3.3',2)
    dataset.append(row)
    target.append(label)

scaler = StandardScaler()
dataset = scaler.fit_transform(dataset)
X_train, X_test, y_train, y_test =
    train_test_split(dataset,target,test_size=0.3,random_state =
    77145416,stratify = target)
y_train = np.array(y_train)
pca = PCA(n_components = 0.95, svd_solver = 'full')
pca.fit(X_train)
X_train = pca.transform(X_train)
X_test = pca.transform(X_test)
clf = 0
for score in scores:
    clf = GridSearchCV(SVC(), param_grid=param_grid, cv=10,
        scoring='%s_macro' % score)
    clf.fit(X_train, y_train)

mean_score_slice = 0
for k in range(0,10):
    indices = np.random.choice(len(X_train),size=len(X_train),replace=False)
    dataset_shuffle = X_train[indices]
    target_shuffle = y_train[indices]
    svm = SVC(C = clf.best_params_['C'],gamma = clf.best_params_['gamma'])
    svm.fit(dataset_shuffle,target_shuffle)
    y_true, y_pred = y_test, svm.predict(X_test)
    mean_score_slice = mean_score_slice + accuracy_score(y_true,y_pred)

mean_score_slice = mean_score_slice/10
accuracy_Zslices.append(mean_score_slice)

h = open("planoZ-d.txt","a")
h.write("Z (Plano 2)")
h.write(str(np.argmax(accuracy_Zslices)))
h.write(str(accuracy_Zslices[np.argmax(accuracy_Zslices)]))
accuracy_Zslices = np.array(accuracy_Zslices)
accuracy_Zslices.tofile('accuracy_Zslices-d.out',sep=",")

```

7.3.1. Resultados experimento 1

```
import numpy as np
```

```

import matplotlib.pyplot as plt

accuracy_X =
    np.loadtxt("accuracy_Xslices-d.out", delimiter=',', dtype=np.float64)
accuracy_Y =
    np.loadtxt("accuracy_Yslices-d.out", delimiter=',', dtype=np.float64)
accuracy_Z =
    np.loadtxt("accuracy_Zslices-d.out", delimiter=',', dtype=np.float64)

mean_accuracy_X = np.mean(accuracy_X)
mean_accuracy_Y = np.mean(accuracy_Y)
mean_accuracy_Z = np.mean(accuracy_Z)

print("Accuracy medio X: ", mean_accuracy_X)
print("Accuracy medio Y: ", mean_accuracy_Y)
print("Accuracy medio Z: ", mean_accuracy_Z)

buenosX = 0
buenosY = 0
buenosZ = 0
slices_bX = []
slices_bY = []
slices_bZ = []
positions_X = []
positions_Y = []
positions_Z = []

for i in range(len(accuracy_X)):
    if(accuracy_X[i] >= 0.75):
        buenosX = buenosX+1
        slices_bX.append(accuracy_X[i])
        positions_X.append(i)

for i in range(len(accuracy_Y)):
    if(accuracy_Y[i] >= 0.75):
        buenosY = buenosY+1
        slices_bY.append(accuracy_Y[i])
        positions_Y.append(i)

for i in range(len(accuracy_Z)):
    if(accuracy_Z[i] >= 0.75):
        buenosZ = buenosZ+1
        slices_bZ.append(accuracy_Z[i])
        positions_Z.append(i)

print("RESULTADOS")
print("PLANO X:")
print("Buenas: ", buenosX)

```

```

print(slices_bX)
print(positions_X)

print("PLANO Y:")
print("Buenas: ", buenasY)
print(slices_bY)
print(positions_Y)

print("PLANO Z:")
print("Buenas: ", buenosZ)
print(slices_bZ)
print(positions_Z)

print("Accuracy para X")

plt.plot(range(0,len(accuracy_X)),accuracy_X.tolist())
plt.xlabel("Slice")
plt.ylabel("Accuracy")
plt.title("Accuracy-Slice en el plano axial")
plt.show()

print("Accuracy para Y")

plt.plot(range(0,len(accuracy_Y)),accuracy_Y.tolist())
plt.xlabel("Slice")
plt.ylabel("Accuracy")
plt.title("Accuracy-Slice en el plano coronal")
plt.show()

print("Accuracy para Z")

plt.plot(range(0,len(accuracy_Z)),accuracy_Z.tolist())
plt.xlabel("Slice")
plt.ylabel("Accuracy")
plt.title("Accuracy-Slice en el plano sagital")
plt.show()

```

7.4. Experimento 2

```

# SCRIPT PARA EL EXPERIMENTO 2
# Sabiendo que el plano coronal ha sido el mas determinante,
# elijo que tipo de imagen da mejores resultados: whole, materia blanco o gris
# Tratamiento por slices
# Author: Luis Balderas Ruiz

```

```

from subjects import Sujeto
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler

nombres_PD = ['./Datos/PD/', './Datos/PD-C1/', './Datos/PD-C2/']
nombres_C = ['./Datos/Control/', './Datos/Control-C1/', './Datos/Control-C2/']

C_range = np.logspace(-2,11,13)
gamma_range = np.logspace(-9,3,13)
#C_range = np.logspace(-5, 15, 11)
#gamma_range = np.logspace(3, -15, 10)
param_grid = dict(gamma=gamma_range, C=C_range)
#param_grid = {'C': [1e3, 5e3, 1e4, 5e4, 1e5],
#              'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1], }

scores = ['recall']

print("LECTURA DE IMAGENES")

for i in range(len(nombres_PD)):

    pacientes = []
    imagenes_pd = os.listdir(nombres_PD[i])

    print("Leyendo pacientes enfermos ", nombres_PD[i])

    for j in range(len(imagenes_pd)):
        imagenes_pd[j] = nombres_PD[i] + imagenes_pd[j]
        paciente = Sujeto(imagenes_pd[j],0)
        pacientes.append(paciente)

    imagenes_control = os.listdir(nombres_C[i])

    print("Leyendo pacientes control")

    for j in range(len(imagenes_control)):
        imagenes_control[j] = nombres_C[i] + imagenes_control[j]
        paciente = Sujeto(imagenes_control[j],1)
        pacientes.append(paciente)

print("Y (Plano 1)")

```

```

Y_shape = pacientes[0].get_shape(1)
accuracy_Yslices = []
components_Yslices = []

for k in range(Y_shape):
    print("Slice ", k)
    dataset = []
    target = []
    np.random.shuffle(pacientes)
    for l in range(len(pacientes)):
        slice = pacientes[l].get_slice(1,k)
        label = pacientes[l].get_label()
        row = pacientes[l].get_wave2D(slice,'bior3.3',2)
        dataset.append(row)
        target.append(label)

scaler = StandardScaler()
dataset = scaler.fit_transform(dataset)
X_train, X_test, y_train, y_test =
    train_test_split(dataset,target,test_size=0.3,random_state =
    77145416, stratify = target)
y_train = np.array(y_train)
pca = PCA(n_components = 0.95, svd_solver = 'full')
pca.fit(X_train)
components_Yslices.append(pca.n_components_)
X_train = pca.transform(X_train)
X_test = pca.transform(X_test)
clf = 0
for score in scores:
    clf = GridSearchCV(SVC(kernel='rbf',class_weight='balanced'),
        param_grid=param_grid, cv=10,
        scoring='%s_macro' % score)
    clf.fit(X_train, y_train)

mean_score_slice = 0
for m in range(0,10):
    indices =
        np.random.choice(len(X_train),size=len(X_train),replace=False)
    dataset_shuffle = X_train[indices]
    target_shuffle = y_train[indices]
    svm = SVC(C = clf.best_params_['C'],gamma =
        clf.best_params_['gamma'])
    svm.fit(dataset_shuffle,target_shuffle)
    y_true, y_pred = y_test, svm.predict(X_test)
    mean_score_slice = mean_score_slice + accuracy_score(y_true,y_pred)

mean_score_slice = mean_score_slice/10
accuracy_Yslices.append(mean_score_slice)

```

```

name = nombres_PD[i]
name = name[-3:-1] + "3"
c = "componentsY-" + name + ".txt"
name_ac = name + "accY3.out"
h = open(name,"a")
h.write(name)
h.write(str(np.argmax(accuracy_Yslices)))
h.write(str(accuracy_Yslices[np.argmax(accuracy_Yslices)]))
accuracy_Yslices = np.array(accuracy_Yslices)
components_Yslices = np.array(components_Yslices)
accuracy_Yslices.tofile(name_ac,sep=",")
components_Yslices.tofile(c,sep=",")

```

7.4.1. Resultados experimento 2

```

import numpy as np
import matplotlib.pyplot as plt

accuracy_C1 = np.loadtxt("C13accY3.out",delimiter=',',dtype=np.float64)
accuracy_C2 = np.loadtxt("C23accY3.out",delimiter=',',dtype=np.float64)
accuracy_W = np.loadtxt("PD3accY3.out",delimiter=',',dtype=np.float64)

cC1 = np.loadtxt("componentsY-C13.txt", delimiter=',',dtype=np.int64)
cC2 = np.loadtxt("componentsY-C23.txt", delimiter=',',dtype=np.int64)
cPD = np.loadtxt("componentsY-PD3.txt", delimiter=',',dtype=np.int64)

dif = cC2 - cC1

plt.plot(range(0,len(dif)),dif.tolist())
plt.xlabel("Slice")
plt.ylabel("Diferencia de accuracy")
plt.title("Diferencia de accuracy para la diferencia C2-C1 en cada slice")
plt.plot(range(0,len(dif)),np.zeros(len(dif)))
plt.show()

mean_accuracy_C1 = np.mean(accuracy_C1)
mean_accuracy_C2 = np.mean(accuracy_C2)
mean_accuracy_W = np.mean(accuracy_W)

print(mean_accuracy_C1)
print(mean_accuracy_C2)
print(mean_accuracy_W)

print("Accuracy para X")

plt.plot(range(0,len(accuracy_C1)),accuracy_C1.tolist(), label = "C1")

```

```

plt.plot(range(0,len(accuracy_C2)),accuracy_C2.tolist(), label = "C2")
#plt.plot(range(0,len(accuracy_W)),accuracy_W.tolist(),label = "W")
plt.xlabel("Slice")
plt.ylabel("Accuracy")
plt.title("Accuracy-Slice C1-C2")
plt.legend()
plt.show()

print("Componentes principales")

plt.plot(range(0,len(cC1)),cC1.tolist())
plt.xlabel("Slice")
plt.ylabel("Numero de componentes")
plt.title("Numero de componentes principales por slice materia gris")
plt.show()

plt.plot(range(0,len(cC2)),cC2.tolist())
plt.xlabel("Slice")
plt.ylabel("Numero de componentes")
plt.title("Numero de componentes principales por slice materia blanca")
plt.show()

plt.plot(range(0,len(cPD)),cPD.tolist())
plt.xlabel("Slice")
plt.ylabel("Numero de componentes")
plt.title("Numero de componentes principales por slice cerebro completo")
plt.show()

```

7.5. Experimento 3

```

# SCRIPT PARA EL EXPERIMENTO 3
# Elegido sagital y cerebro completo, llevo a cabo un ensemble learner
# (stacking)
# para elegir las slices mas determinantes en el diagnostico
# Author: Luis Balderas Ruiz

from subjects import Sujeto
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegressionCV

```

```

pacientes = []
imagenes_pd = os.listdir('./Datos/PD')
# Set the parameters by cross-validation

C_range = np.logspace(-2,11,13)
gamma_range = np.logspace(-9,3,13)
#C_range = np.logspace(-5, 15, 11)
#gamma_range = np.logspace(3, -15, 10)
param_grid = dict(gamma=gamma_range, C=C_range)
#param_grid = {'C': [1e3, 5e3, 1e4, 5e4, 1e5], 'gamma': [0.0001, 0.0005, 0.001,
    0.005, 0.01, 0.1], }

scores = ['recall']
clasificadores = []

print("LECTURA DE IMAGENES")

print("Leyendo pacientes enfermos")

for i in range(len(imagenes_pd)):
    imagenes_pd[i] = './Datos/PD/' + imagenes_pd[i]
    paciente = Sujeto(imagenes_pd[i],0)
    pacientes.append(paciente)

pde = len(pacientes)
print("Enfermos: ",pde)
imagenes_control = os.listdir('./Datos/Control')

print("Leyendo pacientes control")

for i in range(len(imagenes_control)):
    imagenes_control[i] = './Datos/Control/' + imagenes_control[i]
    paciente = Sujeto(imagenes_control[i],1)
    pacientes.append(paciente)

sanos = len(pacientes) - pde
print("Sanos: ",sanos)

## REPARTO LOS PACIENTES ENTRE CONJUNTO DE TRAINING Y TEST ESTRATIFICADO PARA
## LAS DOS FASES DEL
## STACKING (70 %-30 %). PARA TRAINING, 114-55. PARA TEST, 50-23 (ENFERMO-SANO).
pacientes_training = []
train_enfermos = 0
train_sanos = 0
test_enf = 0

```

```

test_sanos = 0
pacientes_test = []
np.random.shuffle(pacientes)
for i in range(len(pacientes)):
    if pacientes[i].get_label() == 0:
        if train_enfermos < 114:
            train_enfermos = train_enfermos + 1
            pacientes_training.append(pacientes[i])
        else:
            test_enf = test_enf + 1
            pacientes_test.append(pacientes[i])
    else:
        if train_sanos < 55:
            train_sanos = train_sanos + 1
            pacientes_training.append(pacientes[i])
        else:
            test_sanos = test_sanos + 1
            pacientes_test.append(pacientes[i])

dataset_final = []
target_final = []
Y_shape = pacientes_training[0].get_shape(1)

for i in range(Y_shape):
    print("Slice ", i)
    dataset = []
    # *_t para pacientes del test y configurar el dataset de la segunda parte
    dataset_t = []
    target = []
    target_t = []
    np.random.shuffle(pacientes_training)
    np.random.shuffle(pacientes_test)
    for j in range(len(pacientes_training)):
        slice = pacientes_training[j].get_slice(1,i)
        label = pacientes_training[j].get_label()
        row = pacientes_training[j].get_wave2D(slice,'bior3.3',2)
        dataset.append(row)
        target.append(label)

    for k in range(len(pacientes_test)):
        slice = pacientes_test[k].get_slice(1,i)
        label = pacientes_test[k].get_label()
        row = pacientes_test[k].get_wave2D(slice,'bior3.3',2)
        dataset_t.append(row)
    dataset_t = np.array(dataset_t)
    print(dataset_t.shape)
    print(len(dataset_t))

scaler = StandardScaler()

```

```

dataset = scaler.fit_transform(dataset)
dataset_t = scaler.fit_transform(dataset_t)
X_train = dataset
y_train = target
pca = PCA(n_components = 0.95, svd_solver = 'full')
pca.fit(X_train)
X_train = pca.transform(X_train)
pca2 = PCA(n_components = 0.95, svd_solver = 'full')
pca2.fit(dataset_t)
dataset_t = pca2.transform(dataset_t)
for score in scores:
    clf = GridSearchCV(SVC(kernel='rbf', class_weight =
        'balanced'), param_grid = param_grid, cv=10, scoring = '%s_macro' %
        score)
    clf.fit(X_train,y_train)
    pred = clf.predict(dataset_t)
    dataset_final.append(pred)

for i in range(len(pacientes_test)):
    target_final.append(pacientes_test[i].get_label())

target_final = np.array(target_final)
dataset_final = np.array(dataset_final)
dataset_final = dataset_final.transpose()
X_train, X_test, y_train, y_test =
    train_test_split(dataset_final,target_final, test_size = 0.2,stratify =
    target_final)
lr = LogisticRegressionCV(cv=10,multi_class='multinomial').fit(X_train,y_train)
sc_training = lr.score(X_train,y_train)
sc_test = lr.score(X_test,y_test)
params_lr = lr.get_params()
name_scores = "scores.txt"
name_params = "params.txt"
name_prob = "probs.txt"

scores = open(name_scores,"a")
scores.write("Score para training: ")
scores.write(str(sc_training))
scores.write("\n")
scores.write("Score para test: ")
scores.write(str(sc_test))
scores.close()

params = open(name_params,"a")
params.write(str(params_lr))
params.close()

mat = lr.coef_

```

```

with open(name_prob,'w') as f:
    for line in mat:
        f.write(str(line))
        f.write("\n")

```

7.5.1. Resultados experimento 3

```

import numpy as np
from numpy import loadtxt

lines = loadtxt("probs.txt",delimiter=",")
mayores = []
menores = []
for i in range(0,10):
    mayor = np.argmax(lines)
    menor = np.argmin(lines)
    mayores.append(mayor)
    menores.append(menor)
    lines[mayor] = 0
    lines[menor] = 0

print("Posiciones mayores")
print(mayores)
print("Posicion menores")
print(menores)

import nibabel as nib
import matplotlib.pyplot as plt

def show_slices(slices):
    """ Function to display row of image slices """
    fig, axes = plt.subplots(1, len(slices))
    for i, slice in enumerate(slices):
        axes[i].imshow(slice.T, cmap="gray", origin="lower")
        if i==1:
            for j in range(len(menores)):
                axes[i].plot([0,150],[mayores[j],mayores[j]],'C3')
        if i==0:
            for j in range(len(menores)):
                axes[i].plot([mayores[j],mayores[j]],[0,150],'C3')

img = nib.load('wSAG_FSPGR_3D_SAG_FSPGR_3D_20130502124054_6.nii')
img_data = img.get_fdata()
print(img_data.shape)

slice_0 = img_data[50,:,:]
slice_1 = img_data[:, :, 50]

```

```
show_slices([slice_0,slice_1])
plt.show()
```

7.5.2. Clasificación con los mejores cortes. Experimento 3

```
# SCRIPT PARA EL EXPERIMENTO 3. Clasificacion posterior
# Elegido sagital y cerebro completo, llevo a cabo un ensemble learner
# (stacking)
# para elegir las slices mas determinantes en el diagnostico
# Comprobando resultados de los 10 mejores cortes
# Author: Luis Balderas Ruiz

from subjects import Sujeto
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(y_true, y_pred,
    classes,normalize=False,title=None,cmap=plt.cm.Blues):
    """
    @brief Funcion encargada de computar y preparar la impresion de la matriz
    de confusión. Se puede extraer los resultados normalizados o sin
    normalizar. Basada en un ejemplo de scikit-learn
    @param y_true Etiquetas verdaderas
    @param y_pred Etiquetas predichas
    @param classes Distintas clases del problema (vector)
    @param normalize Booleano que indica si se normalizan los resultados o no
    @param title Titulo del grafico
    @param cmap Paleta de colores para el grafico
    """
    if not title:
        if normalize:
            title = 'Matriz de confusión normalizada'
        else:
            title = 'Matriz de confusión sin normalizar'

    # Matriz de confusión
    cm = confusion_matrix(y_true, y_pred)
```

```

# Clases
classes = [0,1,2,3,4,5,6,7,8,9]
if normalize:
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    print("Matriz de confusión normalizada")
else:
    print('Matriz de confusión sin normalizar')

print(cm)

fig, ax = plt.subplots()
im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
ax.figure.colorbar(im, ax=ax)
ax.set(xticks=np.arange(cm.shape[1]),
       yticks=np.arange(cm.shape[0]),
       xticklabels=classes, yticklabels=classes,
       title=title,
       ylabel='Etiquetas verdaderas',
       xlabel='Etiquetas predichas')

# Rotar las etiquetas para su posible lectura
plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
          rotation_mode="anchor")

# Creación de anotaciones
fmt = '.2f' if normalize else 'd'
thresh = cm.max() / 2.
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(j, i, format(cm[i, j], fmt),
                ha="center", va="center",
                color="white" if cm[i, j] > thresh else "black")
fig.tight_layout()
return ax

pacientes = []
imagenes_pd = os.listdir('./Datos/PD')
# Set the parameters by cross-validation

C_range = np.logspace(-2,11,13)
gamma_range = np.logspace(-9,3,13)
#C_range = np.logspace(-5, 15, 11)
#gamma_range = np.logspace(3, -15, 10)
param_grid = dict(gamma=gamma_range, C=C_range)
#param_grid = {'C': [1e3, 5e3, 1e4, 5e4, 1e5], 'gamma': [0.0001, 0.0005, 0.001,
#               0.005, 0.01, 0.1], }

scores = ['precision']

```

```

clasificadores = []

print("LECTURA DE IMAGENES")

print("Leyendo pacientes enfermos")

for i in range(len(imagenes_pd)):
    imagenes_pd[i] = './Datos/PD/' + imagenes_pd[i]
    paciente = Sujeto(imagenes_pd[i],0)
    pacientes.append(paciente)

pde = len(pacientes)
print("Enfermos: ",pde)
imagenes_control = os.listdir('./Datos/Control')

print("Leyendo pacientes control")

for i in range(len(imagenes_control)):
    imagenes_control[i] = './Datos/Control/' + imagenes_control[i]
    paciente = Sujeto(imagenes_control[i],1)
    pacientes.append(paciente)

best_slices = [11, 1, 78, 162, 27, 50, 149, 82, 183, 23]

for i in best_slices:
    print("Slice ", i)
    dataset = []
    # *_t para pacientes del test y configurar el dataset de la segunda parte
    dataset_t = []
    target = []
    target_t = []
    np.random.shuffle(pacientes)
    np.random.shuffle(pacientes)
    for j in range(len(pacientes)):
        slice = pacientes[j].get_slice(1,i)
        label = pacientes[j].get_label()
        row = pacientes[j].get_wave2D(slice,'bior3.3',2)
        dataset.append(row)
        target.append(label)

scaler = StandardScaler()
dataset = scaler.fit_transform(dataset)
X_train, X_test, y_train, y_test =
    train_test_split(dataset,target,stratify=target,test_size=0.3,random_state=77145416)
pca = PCA(n_components = 0.95, svd_solver = 'full')
pca.fit(X_train)
X_train_pca = pca.transform(X_train)

```

```

pca2 = PCA(n_components = 0.95, svd_solver = 'full')
pca2.fit(X_test)
X_test_pca = pca.transform(X_test)
for score in scores:
    clf = GridSearchCV(SVC(kernel='rbf', class_weight = 'balanced'), param_grid =
        param_grid, cv=10, scoring = '%s_macro' % score)
    clf.fit(X_train,y_train)
    pred = clf.predict(X_test)
    print(clf.score(X_test,y_test))
    # Pintamos la matriz de confusión
    nombres = ["PD","Control"]
    plot_confusion_matrix(y_test, pred, classes=nombres,normalize =
        False,title='Matriz de confusión')
    plt.show()

    # Pintamos la curva ROC
    print("Área bajo la curva ROC")
    fpr, tpr, threshold = metrics.roc_curve(y_test,pred)
    roc_auc = metrics.auc(fpr,tpr)
    plt.title('Curva ROC')
    plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
    plt.legend(loc = 'lower right')
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlim([0, 1])
    plt.ylim([0, 1])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()

```

7.6. Experimento 4

```

# SCRIPT PARA EL EXPERIMENTO 4
# Elegido sagital y cerebro completo, llevo a cabo un ensemble learner
# (stacking)
# para elegir las slices mas determinantes en el diagnostico
# Enfoque con CNN
# Author: Luis Balderas Ruiz

from subjects import Sujeto
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegressionCV
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout

```

```

from keras.utils import to_categorical
from keras.callbacks import EarlyStopping

pacientes = []
imagenes_pd = os.listdir('./Datos/PD')
# Set the parameters by cross-validation

def deep_learning(X_train,y_train,X_test,y_test):
    model = Sequential()

    #add model layers
    model.add(Conv2D(16, kernel_size=3, activation='relu',
                    input_shape=(157,136,1)))
    model.add(MaxPooling2D(pool_size=(4, 4))) # 2,2 -> 4,4
    model.add(Dropout(0.5))
    model.add(Conv2D(16, kernel_size=3, activation='relu'))
    model.add(MaxPooling2D(pool_size=(4, 4))) # 2,2 -> 4,4
    model.add(Dropout(0.5))
    model.add(Conv2D(16, kernel_size=3, activation='relu'))
    model.add(MaxPooling2D(pool_size=(4, 4))) # 2,2 -> 4,4
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(16, activation='relu'))
    model.add(Dense(2, activation='softmax'))
    #compile model using accuracy as a measure of model performance
    model.compile(optimizer='adam', loss='categorical_crossentropy',
                  metrics=['accuracy'])

    #print(model.summary())
    #train model
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=100,
              callbacks=[EarlyStopping(monitor='val_loss')]) )
    pred = model.predict(X_test)
    pred_fin = []
    for i in range(len(pred)):
        index = pred[i].argmax()
        pred_fin.append(index)
    return pred_fin

clasificadores = []

print("LECTURA DE IMAGENES")

print("Leyendo pacientes enfermos")

```

```

for i in range(len(imagenes_pd)):
    imagenes_pd[i] = './Datos/PD/' + imagenes_pd[i]
    paciente = Sujeto(imagenes_pd[i],0)
    pacientes.append(paciente)

imagenes_control = os.listdir('./Datos/Control')

print("Leyendo pacientes control")

for i in range(len(imagenes_control)):
    imagenes_control[i] = './Datos/Control/' + imagenes_control[i]
    paciente = Sujeto(imagenes_control[i],1)
    pacientes.append(paciente)

## REPARTO LOS PACIENTES ENTRE CONJUNTO DE TRAINING Y TEST ESTRATIFICADO PARA
## LAS DOS FASES DEL
## STACKING (70%-30%). PARA TRAINING, 114-55. PARA TEST, 50-23 (ENFERMO-SANO).
pacientes_training = []
train_enfermos = 0
train_sanos = 0
test_enf = 0
test_sanos = 0
pacientes_test = []
np.random.shuffle(pacientes)
for i in range(len(pacientes)):
    if pacientes[i].get_label() == 0:
        if train_enfermos < 114:
            #if train_enfermos < 8:
            train_enfermos = train_enfermos + 1
            pacientes_training.append(pacientes[i])
        else:
            test_enf = test_enf + 1
            pacientes_test.append(pacientes[i])
    else:
        #if train_sanos < 5:
        if train_sanos < 55:
            train_sanos = train_sanos + 1
            pacientes_training.append(pacientes[i])
        else:
            test_sanos = test_sanos + 1
            pacientes_test.append(pacientes[i])

dataset_final = []
target_final = []
Y_shape = pacientes_training[0].get_shape(1)

for i in range(Y_shape):
    print("Slice ", i, "/", Y_shape)

```

```

dataset = []
# *_t para pacientes del test y configurar el dataset de la segunda parte
dataset_t = []
target = []
target_t = []
np.random.shuffle(pacientes_training)
np.random.shuffle(pacientes_test)
for j in range(len(pacientes_training)):
    slice = pacientes_training[j].get_slice(1,i)
    label = pacientes_training[j].get_label()
    dataset.append(slice)
    target.append(label)

for k in range(len(pacientes_test)):
    slice = pacientes_test[k].get_slice(1,i)
    label = pacientes_test[k].get_label()
    dataset_t.append(slice)

dataset_t = np.array(dataset_t)

X_train = dataset
y_train = target
X_train = np.array(X_train)
y_train = np.array(y_train)
X_train = X_train.reshape(169,157,136,1)
#X_train = X_train.reshape(13,157,136,1)
dataset_t = dataset_t.reshape(73,157,136,1)
#dataset_t = dataset_t.reshape(12,157,136,1)
y_train = to_categorical(y_train)
target_final = []
for i in range(len(pacientes_test)):
    target_final.append(pacientes_test[i].get_label())
target_final = np.array(target_final)
target_final = to_categorical(target_final)
pred = deep_learning(X_train,y_train,dataset_t,target_final)
dataset_final.append(pred)

dataset_final = np.array(dataset_final)
dataset_final = dataset_final.transpose()
etiquetas = []
for i in range(len(target_final)):
    index = target_final[i].argmax()
    etiquetas.append(index)

etiquetas = np.array(etiquetas)
X_train, X_test, y_train, y_test = train_test_split(dataset_final,etiquetas,
    test_size = 0.2,stratify = etiquetas)

```

```

lr = LogisticRegressionCV(cv=10,multi_class='multinomial').fit(X_train,y_train)
sc_training = lr.score(X_train,y_train)
sc_test = lr.score(X_test,y_test)
params_lr = lr.get_params()
name_scores = "scores.txt"
name_params = "params.txt"
name_prob = "probs.txt"

scores = open(name_scores,"a")
scores.write("Score para training: ")
scores.write(str(sc_training))
scores.write("\n")
scores.write("Score para test: ")
scores.write(str(sc_test))
scores.close()

params = open(name_params,"a")
params.write(str(params_lr))
params.close()

mat = lr.coef_
with open(name_prob,'w') as f:
    for line in mat:
        f.write(str(line))

```

7.6.1. Resultados experimento 4

```

import numpy as np
from numpy import loadtxt

lines = loadtxt("probs.txt",delimiter=",")
print(lines)
mayores = []
menores = []
for i in range(0,10):
    mayor = np.argmax(lines)
    menor = np.argmin(lines)
    mayores.append(mayor)
    menores.append(menor)
    lines[mayor] = 0
    lines[menor] = 0

print("Posiciones mayores")
print(mayores)
print("Posicion menores")
print(menores)

```

```

import nibabel as nib
import matplotlib.pyplot as plt

def show_slices(slices):
    """ Function to display row of image slices """
    fig, axes = plt.subplots(1, len(slices))
    for i, slice in enumerate(slices):
        axes[i].imshow(slice.T, cmap="gray", origin="lower")
        if i==1:
            for j in range(len(menores)):
                axes[i].plot([0,150],[menores[j],menores[j]],'C3')
        if i==0:
            for j in range(len(menores)):
                axes[i].plot([menores[j],menores[j]],[0,150],'C3')

img = nib.load('wSAG_FSPGR_3D_SAG_FSPGR_3D_20130503105457_3.nii')
img_data = img.get_fdata()
print(img_data.shape)

slice_0 = img_data[50,:,:]
slice_1 = img_data[:, :, 50]
show_slices([slice_0,slice_1])
plt.show()

```

7.6.2. Clasificación con los mejores cortes. Experimento 4

```

# SCRIPT PARA EL EXPERIMENTO 4
# Elegido sagital y cerebro completo, llevo a cabo un ensemble learner
# (stacking)
# para elegir las slices mas determinantes en el diagnostico. Enfoque con CNN
# Clasificacion posterior
# Author: Luis Balderas Ruiz

from subjects import Sujeto
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping

```

```

def deep_learning(X_train,y_train,X_test,y_test):
    model = Sequential()

    #add model layers
    model.add(Conv2D(16, kernel_size=3, activation='relu',
                    input_shape=(157,136,1)))
    model.add(MaxPooling2D(pool_size=(4, 4))) # 2,2 -> 4,4
    model.add(Dropout(0.5))
    model.add(Conv2D(16, kernel_size=3, activation='relu'))
    model.add(MaxPooling2D(pool_size=(4, 4))) # 2,2 -> 4,4
    model.add(Dropout(0.5))
    model.add(Conv2D(16, kernel_size=3, activation='relu'))
    model.add(MaxPooling2D(pool_size=(4, 4))) # 2,2 -> 4,4
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(16, activation='relu'))
    model.add(Dense(2, activation='softmax'))

    #compile model using accuracy as a measure of model performance
    model.compile(optimizer='adam', loss='categorical_crossentropy',
                  metrics=['accuracy'])

    #print(model.summary())
    #train model
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=100,
              callbacks=[EarlyStopping(monitor='val_loss')])

    pred = model.predict(X_test)
    pred_fin = []
    for i in range(len(pred)):
        index = pred[i].argmax()
        pred_fin.append(index)
    return pred_fin

def plot_confusion_matrix(y_true, y_pred,
                          classes,normalize=False,title=None,cmap=plt.cm.Blues):
    """
    @brief Funcion encargada de computar y preparar la impresion de la matriz
           de confusión. Se puede extraer los resultados normalizados o sin
           normalizar. Basada en un ejemplo de scikit-learn
    @param y_true Etiquetas verdaderas
    @param y_pred Etiquetas predichas
    @param classes Distintas clases del problema (vector)
    @param normalize Booleano que indica si se normalizan los resultados o no
    @param title Titulo del grafico
    @param cmap Paleta de colores para el grafico
    """

    if not title:
        if normalize:

```

```

        title = 'Matriz de confusion normalizada'
    else:
        title = 'Matriz de confusion sin normalizar'

# Matriz de confusion
cm = confusion_matrix(y_true, y_pred)
# Clases
classes = [0,1,2,3,4,5,6,7,8,9]
if normalize:
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    print("Matriz de confusion normalizada")
else:
    print('Matriz de confusion sin normalizar')

print(cm)

fig, ax = plt.subplots()
im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
ax.figure.colorbar(im, ax=ax)
ax.set(xticks=np.arange(cm.shape[1]),
       yticks=np.arange(cm.shape[0]),
       xticklabels=classes, yticklabels=classes,
       title=title,
       ylabel='Etiquetas verdaderas',
       xlabel='Etiquetas predichas')

# Rotar las etiquetas para su posible lectura
plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
         rotation_mode="anchor")

# Creacion de anotaciones
fmt = '.2f' if normalize else 'd'
thresh = cm.max() / 2.
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(j, i, format(cm[i, j], fmt),
                ha="center", va="center",
                color="white" if cm[i, j] > thresh else "black")
fig.tight_layout()
return ax

pacientes = []
imagenes_pd = os.listdir('./Datos/PD')
# Set the parameters by cross-validation

C_range = np.logspace(-2,11,13)
gamma_range = np.logspace(-9,3,13)
#C_range = np.logspace(-5, 15, 11)

```

```

#gamma_range = np.logspace(3, -15, 10)
param_grid = dict(gamma=gamma_range, C=C_range)
#param_grid = {'C': [1e3, 5e3, 1e4, 5e4, 1e5], 'gamma': [0.0001, 0.0005, 0.001,
    0.005, 0.01, 0.1], }

scores = ['precision']
clasificadores = []

print("LECTURA DE IMAGENES")

print("Leyendo pacientes enfermos")

for i in range(len(imagenes_pd)):
    imagenes_pd[i] = './Datos/PD/' + imagenes_pd[i]
    paciente = Sujeto(imagenes_pd[i],0)
    pacientes.append(paciente)

imagenes_control = os.listdir('./Datos/Control')

print("Leyendo pacientes control")

for i in range(len(imagenes_control)):
    imagenes_control[i] = './Datos/Control/' + imagenes_control[i]
    paciente = Sujeto(imagenes_control[i],1)
    pacientes.append(paciente)

best_slices = [58, 145, 79, 64, 105, 180, 54, 30, 12, 117]

for i in best_slices:
    print("Slice ", i)
    dataset = []
    # *_t para pacientes del test y configurar el dataset de la segunda parte
    target = []
    np.random.shuffle(pacientes)
    for j in range(len(pacientes)):
        slice = pacientes[j].get_slice(1,i)
        label = pacientes[j].get_label()
        dataset.append(slice)
        target.append(label)

X_train, X_test, y_train, y_test =
    train_test_split(dataset,target,stratify=target,test_size=0.3,random_state=77145416)
X_train = np.array(X_train)
X_test = np.array(X_test)
y_train = np.array(y_train)
y_test = np.array(y_test)
X_train = X_train.reshape(len(X_train),157,136,1)

```

```

X_test = X_test.reshape(len(X_test),157,136,1)
y_train = to_categorical(y_train)
y_test1 = to_categorical(y_test)
pred = deep_learning(X_train,y_train,X_test,y_test1)

nombres = ["PD","Control"]
plot_confusion_matrix(y_test, pred, classes=nombres,normalize =
    False,title='Matriz de confusión')
plt.show()

# Pintamos la curva ROC
print("Area bajo la curva ROC")
fpr, tpr, threshold = metrics.roc_curve(y_test,pred)
roc_auc = metrics.auc(fpr,tpr)
plt.title('Curva ROC')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()

```

8. Bibliografía

Referencias

- [1] Miriam Mengíbar Rodríguez. Desarrollo de un Sistema de Búsqueda de Imágenes basado en Términos Lingüísticos. 2018.
- [2] Hastie Friedman, Tibshirani. *The Elements of Statistical Learning*. Springer, 2015.
- [3] <https://parkinson.org/>.
- [4] <http://www.parkinson.org/understanding-parkinsons/what-is-parkinsons>.
- [5] Jodi Kashmere, Richard Camicioli, and Wayne Martin. Parkinsonian syndromes and differential diagnosis. *Curr Opin Neurol*, (15):461–466, 2002.
- [6] N. Aggarwal, B. Rana, and R.K. Agrawal. 3d discrete wavelet transform for computer aided diagnosis of Alzheimer's disease using t1-weighted brain MRI. *International Journal of Imaging Systems and Technology*, (25(2)):179–190, 2015.
- [7] Iman Beheshti and Hasan Demirel. Feature-ranking-based Alzheimer's disease classification from structural MRI.
- [8] Deepa Subramaniam Nachimuthu and Arunadevi Baladhandapani. Multidimensional texture characterization: On analysis for brain tumor tissues using MRS and MRI. *Journal of Digital Imaging*, (27(4)):496–506, 2014.
- [9] Mohd Shafry Mohd Rahim, Tanzila Saba, Fatima Nayer, and Afraz Zahra Syed. 3D Texture Features Mining for MRI Brain Tumor Identification. *3D Research*, (5(1)):1–8, 2014.
- [10] R Rajesh Sharma and P Marikkannu. Hybrid RGSA and Support Vector Machine Framework for Three-Dimensional Magnetic Resonance Brain Tumor Classification. 2015.
- [11] Michel E. Vandenberghé, Anne-Sophie Hérard, Nicolas Souedet, Elmahdi Sadouni, Mathieu D.Santin, Dominique Briet, Denis Carré, Jocelyne Schulz, Philippe Hantraye, Pierre-Etienne Chabrier, Thomas Rooney, Thomas Debeir, Véronique Blanchard, Laurent Pradier, Marc Dhenain, and Thierry Delzescaux. High-throughput 3D whole-brain quantitative histopathology in rodents. *Scientific Reports*, (6), 2015.
- [12] Jing Zhang, Chunshui Yu, Guilian Jiang, Weifang Liu, and Longzheng Tong. 3D texture analysis on MRI images of Alzheimer's disease. *Brain Imaging and Behavior*, (6(1)):61–69, 2012.
- [13] Yudong Zhang Zhengchao Dong, Preetha Phillips, Shuihua Wang, Genlin Ji, Ji quan Yang, and Ti-Fei Yuan. Detection of subjects and brain regions related to Alzheimer's disease using 3D MRI scans based on eigenbrain and machine learning. *Frontiers in Computational Neuroscience*, (9):1–15, 2015.

- [14] Olga Valenzuela, Xiaoyi Jiang, Antonio Carrillo, and Ignacio Rojas. Multi-objective genetic algorithms to find most relevant volumes of the brain related to alzheimer's disease and mild cognitive impairment. *International Journal of Neural Systems*, 28(09):1850022, 2018. PMID: 29914313.
- [15] Alberto García Salguero. Use of intelligent systems in the automatic diagnosis of neurodegenerative diseases using medical images. Master Thesis. 2018.
- [16] <https://www.britannica.com/science/nuclear-magnetic-resonance>.
- [17] David Griffiths. *Introduction to Quantum Mechanichs*. 2nd Ed. 2005.
- [18] <https://www.fil.ion.ucl.ac.uk/spm/>.
- [19] K. Bredies and D. Lorenz. *Mathematical Image Processing*. 2018.
- [20] A. Cañada Villar. Series de Fourier y Aplicaciones. *Ed. Pirámide*, 2002.
- [21] I. Daubechies. Ten lectures on Wavelets. *Society for Industrial and Applied Mathematics*, 2006.
- [22] R.M. De Castro Fernández. Análisis de la teoría de ondículas orientada a las aplicaciones en ingeniería eléctrica. *E.T.S.I. Industriales-Madrid*, 2002.
- [23] M. Misiti, Y. Misiti, G. Oppenheim, and J.M Poggi. Wavelets and their Applications. *ISTD Ltd*, 2007.
- [24] Jont B. Allen. Short tiem Spectral Analysis, Synthesis and Modification by DFT. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1977.
- [25] K Gröchenig. Foundations of Time-Frequency Analysis. *Birkhäuser*, 2001.
- [26] Y Meyer. Oscillating Patters in Image Processing and Nonlinear Evolution Equations. *AMS*, 2001.
- [27] J Weidmann. Linear Operators in Hilbert Spaces. *Springer*, 1980.
- [28] L. Simar and W.K. Härdle. Applied Multivariate Statistical Analysis. *Springer*, 2015.
- [29] W.K. Härdle and Z. Hlákva. Multivariate Statistics. *Springer*, 2015.
- [30] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.
- [31] University of Standford. Convolutional Neural Networks. <http://cs231n.github.io/convolutional-networks/>. 2018.
- [32] University of Standford. Introduction to Neural Networks. <http://cs231n.github.io/neural-networks-1/>. 2018.

[33] Fernando Berzal. *Redes Neuronales y Deep Learning*. 2017.

[34] François Chollet. *Deep Learning with Python*. 2018.