UNIVERSIDAD DE CANTABRIA

Facultad de Ciencias

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Clasificación de redes cerebrales basada en medidas de centralidad en grafos

Classification of brain networks based on graph centrality measures

Autor

Carlos Velázquez Fernández

Supervisado por

Camilo Palazuelos Calderón

Rafael Duque Medina

Julio 2024

Índice

1. Introducción **6**
1.1. Motivación 6
1.2. Objetivos 6
1.3. Estructura del documento 7

2. Declaración del problema **8**
2.1. Conectómica 8
2.2. Datos 8
2.3. Representación gráfica del cerebro 9
2.4. Formalización 9

3. Conceptos empleados en la investigación **10**
3.1. Metagrafos 10
3.2. Medidas de centralidad 10
3.2.1. Similitud a los metagrafos 10
3.2.2. Intermediación de las aristas 12
3.2.3. Centralidad de cercanía 12
3.2.4. Eigenvectores 12
3.3. Métodos de clasificación 13
3.3.1. Regresión logística 13
3.3.2. Redes neuronales artificiales 14
3.4. Rendimiento del clasificador 15

4. Experimentación **17**
4.1. Entorno de trabajo 17
4.2. Construcción de los grafos 17
4.3. Aproximación a la puntuación de similitud a los metagrafos 18
4.4. Ajuste del *cross-validation* 19
4.5. Ampliación de la θ 20
4.6. Regresión logística como herramienta de clasificación 20
4.7. Aproximación a la intermediación de aristas 25
4.8. Aproximación a la cercanía 28
4.9. Aproximación a los eigenvectores 29
4.10. Redes neuronales artificiales como herramienta de clasificación 30

5. Conclusiones **34**

6. Líneas futuras **36**

Anexo **39**

Índice de figuras

1. Conjuntos de aristas exclusivos de cada clase. 11
2. Arquitectura de una ANN [16]. 14
3. Funciones de activación sigmoide y ReLU [18]. 15
4. Ejemplo de curva ROC típica [20]. 16
5. Ejemplo en donde $SC^{\mathcal{A}}_{G_c} = SC^{\mathcal{B}}_{G_c}$. 21
6. Ejemplo en donde $SC^{\mathcal{A}}_{G_c} > SC^{\mathcal{B}}_{G_c}$ casi independientemente de las aristas de G_C. 21
7. Representación del conjunto de aristas exclusivo de la clase $\mathcal{A}$. 25

Índice de Tablas

1. Cantidad de muestras por grupo. 8
2. Precisión de la clasificación usando la similitud con los metagrafos. 18
3. Precisión de la clasificación usando la similitud con los metagrafos (con *cross-validation*). 19
4. Proporción de aristas comunes entre $MG^{\mathcal{A}}$ y $MG^{\mathcal{B}}$. 21
5. Precisión del modelo de regresión logística vía puntuaciones de similitud. . . 23
6. AUC del modelo de regresión logística vía puntuaciones de similitud aplicando un *stratified CV*. 23
7. AUC del modelo de regresión logística vía puntuaciones de similitud (con aristas exclusivas) aplicando un *stratified CV*. 25
8. AUC del modelo de regresión logística vía *edge betweenness*. 26
9. AUC del modelo de regresión logística vía *edge betweenness* (con aristas exclusivas). 27
10. AUC del modelo de regresión logística vía *edge betweenness* y puntuación de similitud. 28
11. AUC del modelo de regresión logística vía *edge betweenness* y puntuación de similitud (con aristas exclusivas). 28
12. Mejores AUCs de los modelos de regresión logística con *edge betweenness* . 28
13. AUC del modelo de regresión logística vía *closeness*. 29
14. AUCs de los modelos de regresión logística vía eigenvectores. 30
15. Estructura de la ANN. 31
16. AUC de la ANN vıa puntuaciones de similitud. 32
17. AUCs de las ANN vía *edge betweenness*. 33
18. AUC de la ANN vía *closeness*. 33
19. AUCs de la ANN vía eigenvectores. 33
20. Comparación de resultados de la experimentación. 34
21. Mejores AUCs tras la experimentación completa. 35

Índice de algoritmos

1. Clasificación por similitud 18
2. Clasificación por similitud 2 19
3. Regresión logística vía puntuación de similitud 22
4. Regresión logística vía puntuación de similitud (aristas exclusivas) 24
5. Regresión logística vía intermediación de las aristas 26
6. Regresión logística vía intermediación de las aristas y puntuación de similitud 27
7. Regresión logística vía cercanía 29
8. Regresión logística vía eigenvectores 30
9. Red Neuronal Artificial 32

Resumen

Hoy en día, el uso de la informática como herramienta para lograr avances en otras áreas es de vital importancia. Campos como el de la neurociencia requieren de ésta tanto para procesar como para interpretar conjuntos de datos masivos que nos ayuden a resolver los problemas planteados en la actualidad.

El objetivo de este trabajo será el de continuar con la investigación de Lanciano et al. (2020), en busca de métodos óptimos para clasificar redes neuronales biológicas en función de su padecimiento de determinadas condiciones neurológicas (específicamente, la presencia del trastorno del espectro autista). Aplicando la conectómica cerebral, se representarán los cerebros humanos como grafos y se explorarán diversas medidas de centralidad para extraer información sobre ellos. Haciendo uso de estas métricas, se propondrán distintos modelos de clasificación y se experimentará con ellos para estudiar su comportamiento.

Siguiendo una serie de enfoques que nos permitirán entender mejor las características del problema a resolver, se formularán y contrastarán hipótesis realizando múltiples experimentos, finalizando así con una comparativa general de todas las soluciones propuestas a lo largo del trabajo.

Palabras clave: teoría de grafos; conectómica; clasificación; redes neuronales; autismo

Abstract

Nowadays, the use of Computer Science as a tool to achieve advances in other areas is of vital importance. Fields such as Neuroscience require it both to process and interpret massive data sets that help us solve solve the problems currently raised.

The goal of this work will be to continue with the research of Lanciano et al. (2020), in search of optimal methods to classify biological neural networks based on their suffering from certain neurological conditions (specifically, the presence of autism spectrum disorder). Applying brain connectomics, human brains will be represented as graphs and various centrality measures will be explored to extract information about them. Using these metrics, different classification models will be proposed and experimented with to study their behavior.

Following a series of approaches that will allow us to better understand the characteristics of the problem to be solved, hypotheses will be formulated and contrasted by performing multiple experiments, thus ending with a general comparison of all the solutions proposed throughout the work.

Keywords: graph theory; connectomics; classification; neural networks; autism

1. Introducción

1.1. Motivación

La Neurociencia es la disciplina científica que estudia el sistema nervioso y todos sus aspectos (estructura, función, desarrollo ontogenético y filogenético, bioquímica, farmacología y patología, ...) y cómo sus diferentes elementos interactúan, dando lugar a las bases biológicas de la cognición y la conducta [1]. Una mejor comprensión del cerebro humano podría ayudarnos a detectar o discriminar anomalías, abriendo la puerta a numerosos avances en el área de la Medicina. Es por ello que la investigación en éste campo es de vital importancia en la actualidad.

Basándonos en la conectómica cerebral, disciplina que permite representar las interconexiones entre regiones cerebrales como redes o grafos [2], podemos trasladar el problema desde un campo puramente biológico a un campo matemático, permitiéndonos así emplear herramientas como la Informática para el estudio de éstas estructuras. La importancia de éste análisis radica en la posibilidad de conseguir un mejor entendimiento del funcionamiento de los procesos mentales y enfermedades neurológicas.

1.2. Objetivos

Lanciano et al. (2020)[3] proponían en su trabajo un sistema de clasificación de redes cerebrales basado en la extracción de subgrafos de contraste (conjuntos de vértices cuyos subgrafos inducidos son densos en una de las dos clases). Aplicando este método sobre un conjunto de datos compuesto por sujetos con y sin presencia de un determinado trastorno neurológico, evaluaban la eficacia del clasificador y lo comparaban con otros del estado del arte.

Basándose en esta investigación, se buscará el desarrollo de un clasificador binario de redes cerebrales capaz de discriminar entre sujetos con presencia del trastorno del espectro autista o TEA (grupo de trastornos del desarrollo neurológico que se caracterizan por deficiencias en el comportamiento social y comunicativo y por una gama restringida de actividades e intereses [4]) y sujetos neurotípicos (sin presencia del TEA).

Los principales objetivos de este trabajo se pueden resumir de la siguiente forma:

- Proponer modelos de clasificación distintos a los sugeridos por los autores, así como diversas formas de extraer información de los cerebros (aplicando medidas de centralidad sobre los grafos).
- Aplicar estos modelos sobre el *dataset* de sujetos con presencia de TEA y neurotípicos.
- Comprobar el rendimiento de los clasificadores analizando y comparando los resultados obtenidos en las pruebas experimentales.

Se tratará de encontrar patrones que caractericen las redes cerebrales de ambos tipos y que nos permitan construir un clasificador capaz de etiquetarlas con una tasa de acierto razonable.

1.3. Estructura del documento

Con el fin de facilitar la lectura de esta memoria, se ha dividido el documento en seis secciones diferentes.

- Sección 1. Introducción al trabajo y objetivos que se pretenden alcanzar.
- Sección 2. Declaración formal del problema planteado y explicación de los datos con los que se trabaja y su formato.
- Sección 3. Conceptos importantes que serán mencionados en la experimentación. Aquí se hablará del aspecto teórico de diversas medidas de centralidad de los grafos, clasificadores y otras herramientas que se empleen en el desarrollo del trabajo.
- Sección 4. Desarrollo paso a paso de la experimentación realizada. Se explicará la motivación de cada aproximación, así como el algoritmo empleado para llevarla a cabo y los resultados obtenidos.
- Sección 5. Conclusiones extraídas del trabajo.
- Sección 6. Posible trabajo a futuro.

2. Declaración del problema

2.1. Conectómica

Los conectomas son mapas de conexiones del sistema nervioso, tanto físicas (conexiones estructurales) como temporales (activaciones correlacionadas entre sí por intervalos temporales). Debido a las millones de conexiones nerviosas presentes en el sistema, agrupamos las principales zonas del cerebro en Regiones De Interés (*Regions Of Interest* o ROIs).

Para poder analizar el cerebro con precisión se emplean técnicas como la Imagen por Resonancia Magnética (IRM), que emplea el fenómeno de la resonancia magnética nuclear para obtener información de su estructura y composición [5]. Por otro lado, la Imagen por Resonancia Magnética Funcional (IRMf) nos permite observar las regiones cerebrales activas al realizar determinadas tareas [6].

A partir de un conjunto de datos extraídos de IRMfs de individuos de ambos grupos, sujetos que presentan trastorno del espectro autista (*Autism Spectrum Disorder* o ASD) y sujetos neurotípicos (*Typically Developed* or TD), el objetivo consiste en diferenciar y clasificar correctamente nuevos datos en sendas clases.

2.2. Datos

Los datos originales usados para la investigación fueron extraídos del proyecto *Autism Brain Imagine Data Exchange* (ABIDE)[1] y se dividen en los cuatro grupos representados en la Tabla 1:

Grupo	**Descripción**	**TD**	**ASD**
Children	Edad ≤ 9	52	49
Adolescents	Edad entre [15,20]	121	116
Eyesclosed	Ojos cerrados durante el escáner	158	136
Male	Individuos hombres	418	420

Tabla 1: Cantidad de muestras por grupo.

siendo *Children* el grupo de muestras perteneciente a niños, *Adolescents* el grupo conformado por adolescentes, *Eyesclosed* el grupo compuesto por individuos que mantenían los ojos cerrados mientras se tomaban los datos y *Male* el grupo de sujetos hombres.

Estos datos se preprocesaron por Lanciano, Bonchi y Gionis siguiendo la estrategia DPARSF (*Data Processing Assistant for Resting-State fMRI*)[2] y se encuentran disponibles en su página de Github[3].

Para cada individuo se proporciona un fichero de texto indicando las diversas regiones cerebrales y sus conexiones entre sí.

[1] http://preprocessed-connectomes-project.org/abide/index.html
[2] http://preprocessed-connectomes-project.org/abide/dparsf.html
[3] https://github.com/tlancian/contrast-subgraph

2.3. Representación gráfica del cerebro

Tratándose de redes cerebrales con regiones y conexiones definidas, la mejor representación a la que se puede optar son los grafos. Un grafo G consiste en un conjunto de vértices V y aristas E.

Cada muestra consiste en 116 ROIs, que serán los vértices o nodos de nuestros grafos, y sus conexiones. Cabe destacar que las regiones definidas son idénticas para cualquier cerebro, por lo que todos los grafos tendrán el mismo número de vértices y únicamente variarán las conexiones entre ellos. El número de aristas es indefinido, pudiendo ir desde 0 (ninguna conexión) hasta 6670 (grafo completo).

Esta información se proporciona en forma de matrices de adyacencia a partir de las cuales se construirán los grafos no dirigidos correspondientes.

2.4. Formalización

Partimos de un conjunto de datos $\mathcal{D}$ dividido en dos grupos de individuos: un grupo de condición $\mathcal{A} = \{G_1^A, ..., G_n^A\}$ y un grupo de control $\mathcal{B} = \{G_1^B, ..., G_m^B\}$, en donde cada individuo está representado por un grafo o red neuronal $G_i = (V, E_i)$.

El objetivo es inferir un modelo que dado un nuevo grafo $G_y = (V, E_y)$, prediga correctamente la clase a la que pertenece.

3. Conceptos empleados en la investigación

3.1. Metagrafos

Una de las principales herramientas que emplearemos a la hora de construir el clasificador son los *metagrafos*. Llamaremos metagrafo al grafo resumen de cada clase, i.e., el grafo resultante al tomar la unión de las aristas de todos los individuos. En otras palabras, dispondremos de dos metagrafos

$$MG_{\mathcal{A}} = (V, E^{\mathcal{A}}) \ , \quad E^{\mathcal{A}} = \bigcup_{i=0}^{n} E_i^{\mathcal{A}}$$

$$MG_{\mathcal{B}} = (V, E^{\mathcal{B}}) \ , \quad E^{\mathcal{B}} = \bigcup_{i=0}^{m} E_i^{\mathcal{B}}$$

Estos elementos nos ayudarán a buscar diferencias en las conexiones cerebrales entre ambas clases (recordemos que los vértices son idénticos en todas las muestras y únicamente varían las aristas). Observar qué conexiones ocurren en una clase y no ocurren en otra puede ser un punto clave para discriminar a los individuos. Además, disponemos de otro dato importante, que es la proporción de veces que ocurre cada arista en una clase, o lo que es lo mismo, el número de individuos que presentan dichas conexiones dividido por el número total de individuos de la clase [3]. Tomaremos esta información para asignar peso a las aristas de los metagrafos, siendo:

$$w^{\mathcal{A}}(u, v) = \frac{1}{n} |G_i^{\mathcal{A}} \in \mathcal{A} \ s.t. \ (u, v) \in E_i^{\mathcal{A}}|$$

$$w^{\mathcal{B}}(u, v) = \frac{1}{m} |G_i^{\mathcal{B}} \in \mathcal{B} \ s.t. \ (u, v) \in E_i^{\mathcal{B}}|$$

De esta forma, dispondremos de un grafo resumen pesado de cada clase que nos indicarán en qué proporción aparece cada conexión neuronal en cada grupo.

Para comprobar que las anomalías en las muestras no interfieren de manera negativa en los resultados del clasificador, podemos definir un nuevo parámetro θ que usaremos para podar estos metagrafos. Siendo $0 \leq \theta < 1$, esta variable representará el umbral de poda que se aplicará a las aristas en función de su peso. Indicando un valor bajo, haremos que las conexiones que aparezcan pocas veces en una clase $\mathcal{X}$ no sean representadas en $MG^{\mathcal{X}}$. Igualmente, cuanto más se aumente el valor, mayor será el número de aristas que caigan en el umbral y sean podadas, quedándonos así con aquellas que ocurren en la mayoría de individuos de la clase.

3.2. Medidas de centralidad

Con el objetivo de construir el clasificador, se han tomado diversas medidas, tanto de los grafos como de los metagrafos, para usar como predictores. Principalmente:

3.2.1. Similitud a los metagrafos

Una de las primeras aproximaciones que podemos realizar gracias a los metagrafos, es el cómputo de una puntuación de similitud. Para cada grafo a clasificar, calcularemos su similitud con $MG^{\mathcal{A}}$ y $MG^{\mathcal{B}}$ y usaremos el resultado como predictor para el clasificador.

Para realizar este cálculo se han probado dos métodos distintos.

Las primeras puntuaciones o *scores* ($SC^{\mathcal{A}}$ y $SC^{\mathcal{B}}$) se han computado como la suma de pesos de las aristas en común con sendos metagrafos. De esta forma podemos comprobar numéricamente como de parecido es el conjunto de aristas del grafo a clasificar con respecto a los conjuntos de aristas de cada una de las dos clases. Podemos representarlo como

$$SC^{\mathcal{X}}_{G_y} = \sum w^{\mathcal{X}}(u,v) \in E_y$$

en donde el *score* de un grafo $G_y(V, E_y)$ con respecto a una clase $\mathcal{X}$ es igual a la suma de los pesos $w(u,v)$ del grafo $MG^{\mathcal{X}}$ de las aristas comunes con G_y.

La segunda forma de calcular las puntuaciones ha sido observando únicamente las aristas exclusivas de cada metagrafo (Figura 1). En otras palabras, antes de sumar el peso de cada arista, se aplica la condición de que el metagrafo de la clase contraria no contenga dicha arista. Este método funcionará especialmente bien si hay un conjunto de aristas marcado que caracterice a cada clase.

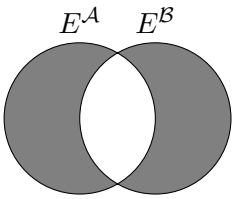


Figura 1: Conjuntos de aristas exclusivos de cada clase.

Esto lo podemos representar como

$$SC^{\mathcal{A}}_{G_y} = \sum w^{\mathcal{A}}(u,v) \in E_y \cap (E^{\mathcal{A}} \setminus E^{\mathcal{B}})$$

$$SC^{\mathcal{B}}_{G_y} = \sum w^{\mathcal{B}}(u,v) \in E_y \cap (E^{\mathcal{B}} \setminus E^{\mathcal{A}})$$

A la hora de interpretar estas puntuaciones, suponemos que la importancia de todas las aristas es la misma y nos guiamos exclusivamente por los pesos. Es decir, en este modelo no contemplamos que aristas o conjuntos de aristas que ocurran pocas veces puedan tener una mayor relevancia a la hora de discriminar entre clases que las aristas que ocurren muchas veces. También cabe destacar que los *scores* por sí mismos no sirven para determinar la clase, puesto que si una clase contiene, en lo general, grafos más densos que la otra, sus puntuaciones correspondientes siempre serán mayores, independientemente de si los grafos a clasificar correspondan o no a ella.

3.2.2. Intermediación de las aristas

Otra medida que podemos usar para caracterizar los grafos es la intermediación de las aristas o *edge betweenness*. La intermediación de una arista e es el número de caminos mínimos que pasan por ella. Si además queremos normalizar los valores, el resultado deberá ser dividido por el número total de caminos mínimos. Siguiendo esta definición, podemos decir que

$$c_B(e) = \sum_{u,v \in V} \frac{\sigma(u, v|e)}{\sigma(u, v)}$$

en donde V es el conjunto de vértices, $\sigma(u, v)$ es el número de caminos mínimos entre los vértices u y v, y $\sigma(u, v|e)$ es el número de caminos mínimos entre u y v que pasan por e [7].

Partiendo de la fórmula del apartado anterior y con el objetivo de encontrar nuevos predictores adecuados para nuestro clasificador, podemos determinar las nuevas puntuaciones de cada grafo a clasificar como

$$SC_{G_y}^{\mathcal{X}} = \sum c_B^{\mathcal{X}}(e) \in E_y$$

en donde el la puntuación de un grafo $G_y(V, E_y)$ con respecto a una clase $\mathcal{X}$ es igual a la suma de los valores de intermediación $c_B(e)$ del grafo $MG^{\mathcal{X}}$ de las aristas comunes con G_y.

Además, repitiendo el razonamiento sobre la exclusividad de las aristas, podemos analizar de nuevo el caso en el que solo tengamos en cuenta las aristas únicas de cada clase, en cuyo caso computaremos:

$$SC_{G_y}^{\mathcal{A}} = \sum c_B^{\mathcal{X}}(e) \in E_y \cap (E^{\mathcal{A}} \setminus E^{\mathcal{B}})$$

$$SC_{G_y}^{\mathcal{B}} = \sum c_B^{\mathcal{X}}(e) \in E_y \cap (E^{\mathcal{B}} \setminus E^{\mathcal{A}})$$

3.2.3. Centralidad de cercanía

La centralidad de cercanía, o *closeness centrality* en inglés, es una métrica usada para medir la distancia desde un nodo hacia todos los demás. La cercanía de un nodo v se calcula como la inversa de la distancia del camino mínimo medio a v desde los $n-1$ nodos alcanzables.

$$C(v) = \frac{n-1}{\sum_{u=1}^{n-1} d(u, v)}$$

en donde $d(u, v)$ es la distancia del camino mínimo desde u hasta v. Si la distancia desde v hacia el resto de nodos del grafo es pequeña, entonces el nodo u tiene una cercanía alta [8][9].

3.2.4. Eigenvectores

Los eigenvectores o vectores propios de un operador lineal son los vectores no nulos que, cuando son transformados por el operador, dan lugar a un múltiplo escalar de sí mismos, con lo que no cambian su dirección. Este escalar λ recibe el nombre de valor propio [10].

La centralidad de vector propio o de eigenvector es una medida de la influencia de un nodo en un grafo. Una puntuación es asignada a cada vértice siguiendo la idea de que no todas sus conexiones son iguales. Las aristas que conectan dicho nodo con otros nodos de puntuación alta, contribuyen a que aumente su propia puntuación y viceversa. En otras palabras, cuanto más influyentes se considere a los vecinos de un nodo (mayor puntuación de vector propio), más influyente se considera al propio nodo [11].

Podemos definir el grado de un nodo v_i del grafo como

$$k_i = \sum_{j=1}^{n} A_{ij}$$

en donde A_{ij} es la matriz de adyacencia del grafo. Si queremos comprobar la centralidad de vector propio, podemos hacer k_i proporcional a la media de las centralidades de los vecinos de v_i:

$$k_i = \frac{1}{\lambda} \sum_{j=1}^{n} A_{ij} x_j$$

en donde λ es un valor constante. Definiendo el vector de centralidades $k = (k_1, k_2, ...)$, podemos reescribir la ecuación en forma matricial como

$$\lambda k = A \cdot k$$

en donde k es el vector propio de la matriz de adyacencia con valor propio λ [12]. Este vector podrá ser utilizado posteriormente como predictor en nuestro clasificador.

3.3. Métodos de clasificación

Una vez expuestas las métricas de centralidad que emplearemos en la investigación, debemos determinar qué método de clasificación usar para discriminar entre las redes cerebrales de ambas clases. Los dos clasificadores que exploraremos en mayor profundidad son los siguientes.

3.3.1. Regresión logística

La regresión logística o modelo logit es un modelo de regresión no linear utilizado cuando la variable dependiente es de tipo dicotómico. El objetivo del modelo es determinar la probabilidad con que una observación pueda generar uno u otro valor de la variable dependiente; pudiéndose usar así para clasificar la observación en dos categorías [13]. Este modelo pertenece a la clase de Modelos Lineares Generalizados (o GLM en inglés).

La ecuación básica para la regresión lineal con variables independientes múltiples es

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_i X_i$$

en donde

- $\hat{Y}$ es el resultado continuo estimado.
- $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_i X_i$ es la ecuación de regresión lineal para las variables independientes del modelo.

- β_0 consiste en la interceptación o punto en el cual la linea de la regresión toca el eje vertical Y (esto es considerado un valor constante).
- $\beta_1 X_1 + \beta_2 X_2 + ... + \beta_i X_i$ es el valor de cada variable independiente (X_i) pesada multiplicada por su coeficiente β.

Los coeficientes beta dan la pendiente a la linea de la regresión. Cuanto mayor sea este valor, mayor es la contribución de su variable independiente correspondiente al resultado [14].

La ecuación de la regresión logística tendrá entonces la siguiente forma

$$\pi = \frac{1}{1 + e^{-\hat{Y}}}$$

siendo el resultado de la función sigmoide una probabilidad entre 0 y 1.

3.3.2. Redes neuronales artificiales

Las redes neuronales artificiales (*Artificial Neural Networks*) son modelos de computación paralelos masivos que imitan el funcionamiento del cerebro humano. Una ANN consiste en un gran número de unidades de proceso (también llamadas neuronas) unidas entre sí por conexiones pesadas, en donde el *output* de cada nodo depende exclusivamente de la información disponible a nivel local por el nodo, ya sea la guardada internamente o la que llega por las conexiones [15]. Esta red aprende de sí misma por experiencia, por lo que no es necesario programarla explícitamente. Su popularidad se debe a que son especialmente buenas dando soluciones a problemas de clasificación, reconocimiento de patrones, procesado de imágenes, etc.

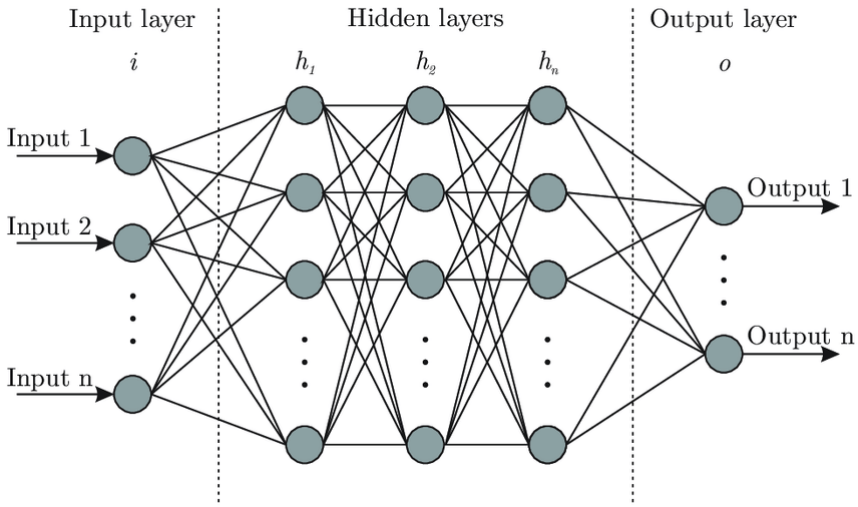


Figura 2: Arquitectura de una ANN [16].

Como se refleja en la Figura 2, la arquitectura general de una red neuronal artificial consiste en una serie de tres o más capas. La primera capa, o capa de entrada i, consta de n neuronas, siendo n el número de variables que definen el dato a clasificar. La capa o capas intermedias h constarán de un número de neuronas indefinido, las cuales variarán el peso de las conexiones a lo largo del aprendizaje. La última capa, o capa de salida o, tendrá m neuronas, siendo m el número de clases a las que pueden pertenecer los datos y

definirá el resultado de la clasificación.

Cada nodo recibe múltiples *inputs* por medio de sus conexiones con otras neuronas. Cuando la suma pesada de éstos supera cierto umbral establecido, la neurona se activa y pasa la señal por una función de activación para posteriormente enviarla a las unidades vecinas. Este proceso se puede modelar como

$$y = f\left(\sum_{i=0}^{n} w_i x_i - T\right)$$

en donde y es el *output* del nodo, f es la función de activación, w_i es el peso del *input* x_i y T es el valor del umbral [17]. Las principales funciones de activación en las que nos centraremos son la función rectificador (ReLU) y la función sigmoide.

La función rectificador se define como

$$f(x) = max(0, x) = \begin{cases} x & si \quad x > 0, \\ 0 & en\ caso\ contrario, \end{cases}$$

Por otro lado, la función sigmoide tiene la forma

$$f(x) = \frac{1}{1 + e^{-x}}$$

siendo x en ambas el *input* de la neurona. Recordemos que la regresión logística (véase en la sección 3.3.1) también hace uso de la función sigmoide. En la Figura 3 se representan gráficamente ambas funciones.

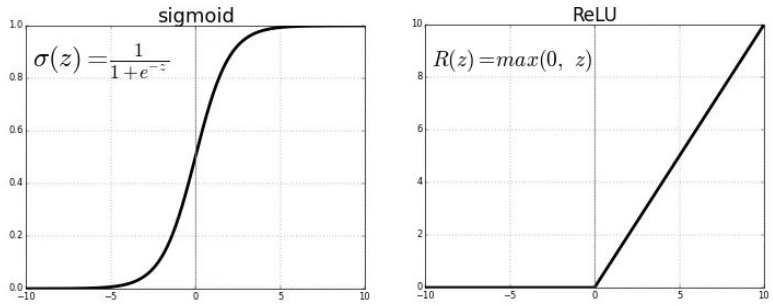


Figura 3: Funciones de activación sigmoide y ReLU [18].

3.4. Rendimiento del clasificador

A la hora de determinar cómo de bueno es nuestro clasificador y con el objetivo de poder compararlo con otros modelos, utilizaremos dos herramientas principalmente: la precisión y el área bajo la curva ROC.

Cuando hablamos de precisión nos referimos a la proporción de datos que el modelo puede clasificar correctamente, sea así

$$p = \frac{aciertos}{aciertos + fallos}$$

siendo p un número entre 0 y 1 (o un porcentaje).

Por otro lado, la curva ROC (del inglés *Receiver Operating Characteristic*) es un gráfico que representa el rendimiento del modelo de clasificación binario. Esta gráfica muestra la tasa de verdaderos positivos (TPR) en contraposición de los falsos positivos (FPR) basándose en las predicciones del modelo. Podemos calcular TPR y FPR como

$$TPR = \frac{TP}{TP + FN} \qquad\qquad FPR = \frac{FP}{FP + TN}$$

siendo TP los verdaderos positivos, FN los falsos negativos, FP los falsos positivos y TN los verdaderos negativos.

Un clasificador ideal debería dar una TPR alta y, por el contrario, una FPR baja, i.e., una identificación correcta de los individuos de una clase sin etiquetar incorrectamente a los de la otra [19]. La métrica que usaremos para determinar la optimalidad de nuestro clasificador será el área bajo la curva ROC, ya que es un único número independiente del algoritmo de clasificación y de la distribución de clases, que nos proporciona suficiente información para entender cómo de bueno es su rendimiento.

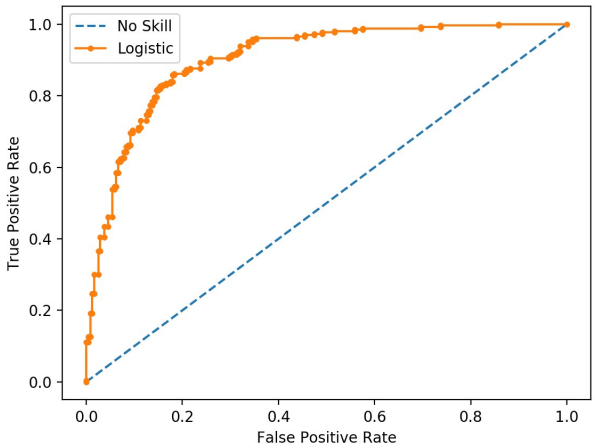


Figura 4: Ejemplo de curva ROC típica [20].

En la Figura 4 podemos observar un ejemplo de curva ROC, i.e, la evolución del TPR con respecto al FPR en distintos umbrales. En azul se representa la clasificación completamente aleatoria (el clasificador no es capaz de discriminar en absoluto). En naranja, la clasificación actual de un modelo que diferencia correctamente los datos en la mayoría de situaciones.

4. Experimentación

4.1. Entorno de trabajo

El *hardware* empleado para el desarrollo y *testing* del programa consiste en el siguiente:

- Sistema operativo: Windows 11 Home de 64 bits.
- Procesador: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz.
- Memoria RAM: 16 GB.

En cuanto al *software* usado:

- Lenguaje de programación: Python 3.11.2 64-bit.
- Principales librerías:
  - Keras 3.3.3.
  - NetworkX 3.3.
  - Numpy 1.26.4.
  - Scikit-learn 1.4.2.
- Entorno de desarrollo: Spyder IDE 5.5.4.

La elección de Python como principal lenguaje para la realización del proyecto viene motivada por la gran disponibilidad de librerías de grafos, minería de datos, estadística y aprendizaje automático que éste presenta.

4.2. Construcción de los grafos

Partimos de un conjunto de datos de cuatro grupos, cada uno de ellos con individuos clasificados en dos clases (*asd* y *td*). Sin embargo, la información que recibimos viene dada en forma de matrices de adyacencia, por lo que nuestro primer paso será transformar ésta en grafos con los que poder trabajar.

A partir de los ficheros de texto con las matrices, construiremos los grafos como objetos de la librería *networkX* de Python. Serán grafos adireccionales (las matrices son simétricas) y no pesados.

Una vez leídos y almacenados todos los grafos, es necesario computar el par de metagrafos correspondiente a cada grupo. Para cada clase construiremos un grafo en donde su conjunto de aristas será la unión de aristas de los grafos de esa clase y cuyos pesos serán la proporción de veces que ocurren esas aristas en los grafos de dicha clase.

Si queremos comprobar qué ocurre en el caso de que suprimamos las aristas que se encuentren por debajo de un umbral θ, es el momento de eliminarlas de nuestros grafos. El resultado serán los metagrafos procesados, listos para trabajar con ellos.

4.3. Aproximación a la puntuación de similitud a los metagrafos

Partiendo de los metagrafos construidos, podemos tratar de realizar una primera clasificación en base al siguiente algoritmo:

Algorithm 1 Clasificación por similitud

$G_c(V, E) \leftarrow$ Grafo a clasificar
$MG^{\mathcal{A}} \leftarrow$ Metagrafo de la clase $\mathcal{A}$
$MG^{\mathcal{B}} \leftarrow$ Metagrafo de la clase $\mathcal{B}$
$SC^{\mathcal{A}} \leftarrow 0$
$SC^{\mathcal{B}} \leftarrow 0$
for $e \in E$ **do** ▷ Calculamos la puntuación de similitud.
    $SC^{\mathcal{A}} \leftarrow SC^{\mathcal{A}} + w^{\mathcal{A}}(e)$
    $SC^{\mathcal{B}} \leftarrow SC^{\mathcal{B}} + w^{\mathcal{B}}(e)$
end for
Predicción $\leftarrow clase(max(SC^{\mathcal{A}}, SC^{\mathcal{B}}))$ ▷ La clase resultante es la de puntuación mayor.

En otras palabras, calculamos la similitud del grafo a clasificar con ambos metagrafos de su grupo siguiendo el procedimiento explicado en el apartado 3.2.1. Este primer modelo nos permite generalizar las características de cada clase y probar una clasificación en base a un criterio muy sencillo (elegir la clase cuya puntuación sea mayor). Tras ejecutar el código para todas las clases y variando el umbral θ en el rango [0, 0.9] con pasos de 0.1, obtendremos los resultados de la Tabla 2.

Meta-graph Only Classification										
Theta	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Adolescents	77.22	82.28	78.48	74.68	67.51	56.54	53.16	51.05	51.05	51.05
Eyesclosed	65.31	66.67	57.14	61.56	54.42	53.74	53.74	53.74	53.74	53.74
Male	58.47	59.31	57.16	56.80	54.18	53.94	51.07	49.88	49.88	49.88
Children	95.05	94.06	94.06	91.09	84.16	61.39	48.51	48.51	55.45	51.49

Tabla 2: Precisión de la clasificación usando la similitud con los metagrafos.

La conclusión que podemos extraer de esta tabla es que el modelo obtiene una buena precisión para los grupos con pocos datos (*Children* y *Adolescents*) y además funciona mejor cuantas menos aristas omitamos, es decir, con $\theta = 0$. Sin embargo, la explicación más sencilla sobre los resultados obtenidos es que nuestro modelo se entrena con los mismos datos que posteriormente usamos para clasificar. Por ello, debemos hacer un pequeño ajuste al algoritmo.

Trataremos de repetir el proceso, pero esta vez creando un nuevo metagrafo por cada uno de nuestros grafos. Estos metagrafos serán muy similares a los anteriores, pero evitarán que el modelo conozca de antemano los datos a clasificar. Cada nuevo $MG_i^{\mathcal{X}}$ contendrá la unión de todas las aristas de los grafos de la clase $\mathcal{X}$ a excepción de aquellas del grafo $G_i^{\mathcal{X}}$. Es decir, a la hora de clasificar $G_i^{\mathcal{X}}$, calcularemos su puntuación con respecto a los metagrafos $MG_i^{\mathcal{X}}$ y $MG^{\mathcal{Y}}$.

Algorithm 2 Clasificación por similitud 2

$G_c(V, E) \leftarrow$ Grafo a clasificar
$MG^{\mathcal{A}} \leftarrow$ Metagrafo de la clase $\mathcal{A}$
$MG^{\mathcal{B}} \leftarrow$ Metagrafo de la clase $\mathcal{B}$
$SC^{\mathcal{A}} \leftarrow 0$
$SC^{\mathcal{B}} \leftarrow 0$
if $G_c \in \mathcal{A}$ **then** ▷ Escogemos los metagrafos que no contengan información sobre el grafo.
    $MG^{\mathcal{A}} \leftarrow MG_c^{\mathcal{A}}$
else
    $MG^{\mathcal{B}} \leftarrow MG_c^{\mathcal{B}}$
end if
for $e \in E$ **do** ▷ Calculamos la puntuación de similitud.
    $SC^{\mathcal{A}} \leftarrow SC^{\mathcal{A}} + w^{\mathcal{A}}(e)$
    $SC^{\mathcal{B}} \leftarrow SC^{\mathcal{B}} + w^{\mathcal{B}}(e)$
end for
Predicción $\leftarrow clase(max(SC^{\mathcal{A}}, SC^{\mathcal{B}}))$ ▷ La clase resultante es la de puntuación mayor.

Con esta validación cruzada conseguiremos unos resultados mucho más fieles a la realidad, representados en Tabla 3.

Meta-graph Only Classification (MG Cross-Validation)										
Theta	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Adolescents	56.96	58.23	56.12	55.70	53.59	53.59	51.05	51.05	51.05	50.63
Eyesclosed	52.72	53.06	53.06	53.74	53.40	53.74	53.74	53.74	53.74	53.74
Male	53.46	54.18	54.77	53.58	53.46	53.70	50.95	49.88	49.88	49.88
Children	53.47	47.52	45.54	38.61	36.63	54.46	48.51	48.51	55.45	51.49

Tabla 3: Precisión de la clasificación usando la similitud con los metagrafos (con *cross-validation*).

Desgraciadamente los resultados en esta ocasión no son tan buenos. Con una precisión media de entre el 50 % y 60 % para las mejores θ, podemos concluir que este modelo no es tan útil como desearíamos. No obstante, esta aproximación será la base de todas las demás, permitiéndonos construir sobre ella aplicando nuevos enfoques para resolver el problema.

4.4. Ajuste del *cross-validation*

Uno de los problemas que surge tras la experimentación expuesta es el coste temporal del algoritmo. Si bien un *leave-one-out cross-validation* nos proporciona un método granular para entrenar nuestro modelo, permitiéndonos tomar un *training set* de $n-1$ elementos y un *testing set* de 1 elemento en un *dataset* de n muestras, esta validación cruzada tan exhaustiva requiere tiempos de ejecución extremadamente elevados.

En caso de querer avanzar hacia un modelo de clasificación más complejo, aparece el inconveniente de que para cada grafo a clasificar se debe computar un metagrafo propio (construido a partir de todos los demás). Para cada nuevo modelo las operaciones a realizar serían las siguientes:

1. Separar el grafo a clasificar G_c.
2. Computar un $MG^{\mathcal{X}}$ para cada $G^{\mathcal{X}}$.

3. Computar $(SC^{\mathcal{A}}, SC^{\mathcal{B}})$ para todos los grafos menos G_c.

4. Entrenar el modelo en base a las puntuaciones y sus etiquetas.

5. Predecir la clase de G_c calculando $(SC_c^{\mathcal{A}}, SC_c^{\mathcal{B}})$.

En conclusión, aunque este método pueda proporcionar una ligera mejora sobre los resultados al crear modelos más completos, en la práctica se trabajará con otro tipo de validación que alcance un punto medio entre la precisión del clasificador y el tiempo necesario para ejecutar el programa: la *stratified cross-validation*.

La validación cruzada estratificada nos permitirá reducir el tamaño del conjunto de entrenamiento aumentado el de testeo. Seleccionando el número de divisiones que deseamos tener en nuestro *dataset* [4], se crearán grupos que contendrán un número similar de muestras de cada clase. Se realizará un bucle de tantas iteraciones como divisiones se hayan establecido, seleccionando en cada una un conjunto de *testing* y los demás de *training*. De esta forma aseguraremos que los modelos siguen siendo robustos a la vez que disminuimos el tiempo para entrenarlos.

4.5. Ampliación de la θ

Hasta ahora hemos probado a establecer un umbral de poda de los metagrafos θ y variarlo para comprobar el comportamiento de nuestro clasificador en cada caso.

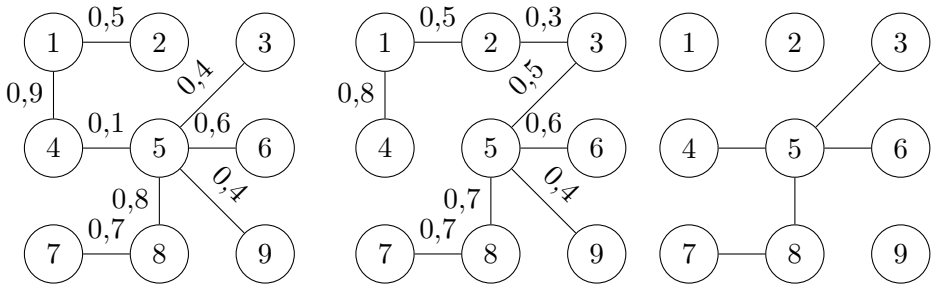
En las siguientes secciones comprobaremos que la θ que mejor rendimiento consigue dependerá de numerosos factores como el grupo de los datos, el método de clasificación o las medidas de centralidad aplicadas. Debido a esto, de ahora en adelante añadiremos un nuevo caso a representar en cada tabla, $\theta = All$.

El significado de $\theta = All$ será el uso de todas las puntuaciones variando la variable θ en el rango [0, 0.9] para clasificar cada grafo. En otras palabras, para clasificar un grafo G, la variable predictiva a computar será una lista de pares de la forma $[(SC_{\theta=0}^{\mathcal{A}}, SC_{\theta=0}^{\mathcal{B}}), ..., (SC_{\theta=0,9}^{\mathcal{A}}, SC_{\theta=0,9}^{\mathcal{B}})]$. De esta forma estaremos utilizando el conjunto completo de información para inferir la clase a la que pertenece nuestro grafo.

4.6. Regresión logística como herramienta de clasificación

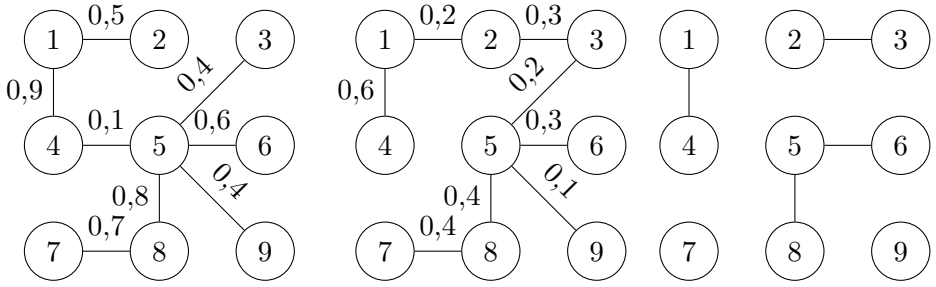
El modelo anterior puede fallar por diversos motivos. Supongamos que los metagrafos $MG^{\mathcal{A}}$ y $MG^{\mathcal{B}}$ tienen un conjunto de aristas y pesos muy similar entre sí, es decir, que la mayoría de conexiones son comunes a ambas clases. En este caso obtendremos puntuaciones de similitud muy parecidas, que complicarán la tarea de clasificación (Figura 5). Por otro lado, si el conjunto de aristas en común es grande, pero una de las dos clases se caracteriza por tener más conexiones en general que la otra, la puntuación de dicha clase tenderá a ser siempre mayor, independientemente de si el grafo a clasificar realmente pertenece a ella (Figura 6).

[4] En el código se emplearán 5 *splits*.



(a) $MG^{\mathcal{A}}$ (b) $MG^{\mathcal{B}}$ (c) G_C

Figura 5: Ejemplo en donde $SC^{\mathcal{A}}_{G_c} = SC^{\mathcal{B}}_{G_c}$.



(a) $MG^{\mathcal{A}}$ (b) $MG^{\mathcal{B}}$ (c) G_C

Figura 6: Ejemplo en donde $SC^{\mathcal{A}}_{G_c} > SC^{\mathcal{B}}_{G_c}$ casi independientemente de las aristas de G_C.

Ambos problemas surgen en caso de que el conjunto de aristas de ambos metagrafos sea muy similar. Si queremos comprobar la veracidad de nuestra hipótesis, solo debemos calcular la proporción de aristas comunes entre los metagrafos (Tabla 4).

Common Edge Proportion										
Theta	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Adolescents	97.38%	80.21%	81.49%	83.07%	83.64%	82.10%	85.07%	84.90%	83.46%	80.72%
Eyesclosed	99.15%	80.68%	81.50%	84.84%	84.39%	84.77%	86.34%	85.86%	88.76%	88.53%
Male	99.82%	87.33%	87.48%	89.09%	87.60%	89.15%	87.70%	87.50%	89.73%	87.72%
Children	92.94%	72.98%	72.40%	75.19%	76.34%	78.92%	80.86%	78.49%	81.36%	78.23%

Tabla 4: Proporción de aristas comunes entre $MG^{\mathcal{A}}$ y $MG^{\mathcal{B}}$.

Con esta tabla podemos extraer la conclusión de que efectivamente nos encontramos en el caso supuesto. Los metagrafos de las dos clases tienen una inmensa cantidad de aristas comunes, especialmente cuando no están podados ($\theta = 0$). Sin embargo, también podemos observar que al eliminar aristas siguiendo el umbral θ, este número disminuye ligeramente. Esto significa que existe un grupo pequeño de conexiones que tienen a ocurrir más en una clase que en otra.

La solución que podemos proponer ante este problema es un nuevo método de clasificación: la regresión. Dado que únicamente necesitamos predecir si un grafo pertenece o no a una clase (0 o 1), nuestra mejor opción será optar por una regresión logística. Usando el par de puntuaciones como predictores, ahora no importará cuál de ellas sea mayor, sino que será la combinación de ambas la que determine la clasificación.

Algorithm 3 Regresión logística vía puntuación de similitud

entrenamiento ← Grafos de entrenamiento
testeo ← Grafos a clasificar
predictoresEntrenamiento ← [ ]
predictoresTesteo ← [ ]
for $G(V, E) \in training \cup testing$ **do**
  $SC^{\mathcal{A}} \leftarrow 0$
  $SC^{\mathcal{B}} \leftarrow 0$
  if $G(V, E) \in training$ **then** ▷ Seleccionamos los metagrafos correspondientes.
    if $G \in \mathcal{A}$ **then** ▷ (a)
      $MG^{\mathcal{A}} \leftarrow$ Metagrafo de la clase $\mathcal{A}$ específico para G
      $MG^{\mathcal{B}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{B}$
    else ▷ (b)
      $MG^{\mathcal{A}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{A}$
      $MG^{\mathcal{B}} \leftarrow$ Metagrafo de la clase $\mathcal{B}$ específico para G
    end if
  else
    $MG^{\mathcal{A}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{A}$
    $MG^{\mathcal{B}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{B}$
  end if
  for $e \in E$ **do** ▷ Calculamos la puntuación de similitud.
    $SC^{\mathcal{A}} \leftarrow SC^{\mathcal{A}} + w^{\mathcal{A}}(e)$
    $SC^{\mathcal{B}} \leftarrow SC^{\mathcal{B}} + w^{\mathcal{B}}(e)$
  end for
  if $G(V, E) \in training$ **then**
    $predictoresEntrenamiento.append((SC^{\mathcal{A}}, SC^{\mathcal{B}}))$
  else
    $predictoresTesteo.append((SC^{\mathcal{A}}, SC^{\mathcal{B}}))$
  end if
end for
RL ← $regresionLogistica(predictoresEntrenamiento, [\mathcal{A}, \mathcal{A}, \mathcal{B}, ..., \mathcal{A}])$ ▷ (c)
Predicciones ← $RL.clase(predictoresTesteo)$ ▷ (d)

(a) Si el grafo es de entrenamiento y pertenece a la clase $\mathcal{A}$, su metagrafo de la clase $\mathcal{A}$ será aquel construido por todos los grafos de entrenamiento de la clase $\mathcal{A}$ menos él mismo.
(b) Si el grafo es de entrenamiento y pertenece a la clase $\mathcal{B}$, su metagrafo de la clase $\mathcal{B}$ será aquel construido por todos los grafos de entrenamiento de la clase $\mathcal{B}$ menos él mismo.
(c) Generamos el modelo de regresión logística a partir de las puntuaciones de todos los grafos menos aquellos que deseamos clasificar y sus respectivas etiquetas.
(d) Predecimos según nuestro modelo dadas las puntuaciones delos grafos a clasificar.

Siguiendo la estructura del Algoritmo 3 para cada iteración de la validación cruzada para clasificar todos los datos de nuestro conjunto empleando un modelo de regresión logística. Los resultados se muestran en la Tabla 5.

Logistic Regression Classification (all edges)										
Theta	**0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Adolescents	59.49	61.18	56.96	58.65	60.76	60.76	55.27	51.05	53.16	54.85
Eyesclosed	55.78	57.14	55.78	54.42	57.48	53.74	50.34	49.66	55.78	54.76
Male	59.79	60.50	60.50	59.07	60.86	59.90	58.11	55.61	57.76	53.58
Children	55.45	57.43	57.43	61.39	53.47	50.50	48.51	50.50	55.45	55.45

Tabla 5: Precisión del modelo de regresión logística vía puntuaciones de similitud.

Podemos notar resultados algo mejores, especialmente para los casos en los que podamos poco los metagrafos (rango [0, 0.5]). Llegados al punto de tener un modelo predictivo más fiable como este, podemos cambiar la representación de su rendimiento. Dejando de lado la precisión, ahora observaremos el área bajo la curva (*Area Under Curve*) ROC del clasificador (Tabla 6). Además, podemos introducir en el experimento el caso $\theta = All$ representado en una nueva columna.

Logistic Regression Classification (all edges)											
Theta	**All**	**0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Adolescents	0.6540	0.6539	0.6491	0.6401	0.6386	0.6528	0.6329	0.5876	0.5402	0.5441	0.5537
Eyesclosed	0.5471	0.5901	0.5925	0.5988	0.5681	0.5763	0.5306	0.5176	0.5296	0.5541	0.5114
Male	0.5926	0.6510	0.6498	0.6509	0.6271	0.6500	0.6328	0.6189	0.5854	0.5788	0.5526
Children	0.6283	0.5863	0.5875	0.5734	0.5824	0.5467	0.5506	0.4937	0.4918	0.5275	0.6013

Tabla 6: AUC del modelo de regresión logística vía puntuaciones de similitud aplicando un *stratified CV*.

Analizando estos resultados, podemos concluir que, por lo general, el modelo predice con mayor precisión cuando trabaja con los metagrafos en los que la θ es menor. Esto podría deberse a que, aunque al podar los metagrafos se crean dos conjuntos de aristas diferenciados (uno para cada clase) (Tabla 4), no es tan común que todos los grafos de la clase compartan entre sí esas aristas exclusivas.

Llegados a este punto, sería interesante comprobar qué ocurriría si, en efecto, dejamos a un lado las aristas comunes entra ambas clases y únicamente tenemos en cuenta aquellas en las que se diferencia. Para ello, modificaremos ligeramente nuestro algoritmo de la siguiente forma:

Algorithm 4 Regresión logística vía puntuación de similitud (aristas exclusivas)

$entrenamiento \leftarrow$ Grafos de entrenamiento
$testeo \leftarrow$ Grafos a clasificar
$predictoresEntrenamiento \leftarrow [\,]$
$predictoresTesteo \leftarrow [\,]$
$MG^{\mathcal{A}} \leftarrow$ Metagrafo de la clase $\mathcal{A}$
$MG^{\mathcal{B}} \leftarrow$ Metagrafo de la clase $\mathcal{B}$
for $G(V, E) \in training \cup testing$ **do**
  $SC^{\mathcal{A}} \leftarrow 0$
  $SC^{\mathcal{B}} \leftarrow 0$
  if $G(V, E) \in training$ **then** ▷ Seleccionamos los metagrafos correspondientes.
    if $G \in \mathcal{A}$ **then**
      $MG^{\mathcal{A}} \leftarrow$ Metagrafo de la clase $\mathcal{A}$ específico para G
      $MG^{\mathcal{B}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{B}$
    else
      $MG^{\mathcal{A}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{A}$
      $MG^{\mathcal{B}} \leftarrow$ Metagrafo de la clase $\mathcal{B}$ específico para G
    end if
  else
    $MG^{\mathcal{A}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{A}$
    $MG^{\mathcal{B}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{B}$
  end if
  for $e \in E$ **do** ▷ Calculamos la puntuación de similitud.
    if $e \in E^{\mathcal{A}} \setminus E^{\mathcal{B}}$ **then** ▷ Solo si e es exclusiva de la clase $\mathcal{A}$.
      $SC^{\mathcal{A}} \leftarrow SC^{\mathcal{A}} + w^{\mathcal{A}}(e)$
    else if $e \in E^{\mathcal{B}} \setminus E^{\mathcal{A}}$ **then** ▷ Solo si e es exclusiva de la clase $\mathcal{B}$.
      $SC^{\mathcal{B}} \leftarrow SC^{\mathcal{B}} + w^{\mathcal{B}}(e)$
    end if
  end for
  if $G(V, E) \in training$ **then**
    $predictoresEntrenamiento.append((SC^{\mathcal{A}}, SC^{\mathcal{B}}))$
  else
    $predictoresTesteo.append((SC^{\mathcal{A}}, SC^{\mathcal{B}}))$
  end if
end for
RL $\leftarrow regresionLogistica(predictoresEntrenamiento, [\mathcal{A}, \mathcal{A}, \mathcal{B}, ..., \mathcal{A}])$
Predicciones $\leftarrow RL.clase(predictoresTesteo)$

De esta forma, la puntuación de similitud con respecto a la clase $\mathcal{A}$ ($SC_c^{\mathcal{A}}$), únicamente estará compuesta por el conjunto de aristas exclusivas de dicha clase (Figura 7) y viceversa.

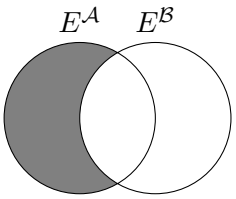


Figura 7: Representación del conjunto de aristas exclusivo de la clase $\mathcal{A}$.

Tras ejecutar el nuevo programa, obtendremos los resultados de la Tabla 7.

Logistic Regression Classification (exclusive edges)											
Theta	**All**	**0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Adolescents	0.6166	0.5387	0.6110	0.6234	0.6227	0.6580	0.6334	0.5837	0.5089	0.5347	0.5564
Eyesclosed	0.6134	0.5226	0.5944	0.6214	0.5749	0.5758	0.5292	0.5065	0.4950	0.5591	0.5079
Male	0.6074	0.5374	0.6143	0.6314	0.5976	0.6439	0.6272	0.6180	0.5774	0.5631	0.5419
Children	0.5958	0.5726	0.5848	0.5553	0.6005	0.5549	0.5498	0.4717	0.4533	0.5216	0.5761

Tabla 7: AUC del modelo de regresión logística vía puntuaciones de similitud (con aristas exclusivas) aplicando un *stratified CV*.

Comparando esta tabla con la Figura 6 en la que teníamos en cuenta todas las aristas (comunes y no comunes), podemos determinar que no hay cambios sustanciales. Los mejores resultados de cada grupo varían ligeramente pero no lo suficiente como para concluir que existe una mejora.

4.7. Aproximación a la intermediación de aristas

Tras haber construido un modelo predictivo que clasifica a los individuos en base su similitud con los metagrafos de cada clase y usando la regresión logística como método de clasificación, es momento de explorar nuevas medidas de centralidad con las que podamos alimentar nuestro programa.

En este caso, incluiremos información en nuestro algoritmo sobre la intermediación de las aristas de los metagrafos. Recordemos que la intermediación de una arista o *edge betweenness* es el número de caminos mínimos que pasan por ella. Este dato podría ser muy útil para identificar conexiones que, aunque ocurran pocas veces, sean relevantes a la hora de explorar el grafo. Modificando nuestro algoritmo:

Algorithm 5 Regresión logística vía intermediación de las aristas

$entrenamiento \leftarrow$ Grafos de entrenamiento
$testeo \leftarrow$ Grafos a clasificar
$predictoresEntrenamiento \leftarrow [\,]$
$predictoresTesteo \leftarrow [\,]$
$MG^{\mathcal{A}} \leftarrow$ Metagrafo de la clase $\mathcal{A}$
$MG^{\mathcal{B}} \leftarrow$ Metagrafo de la clase $\mathcal{B}$
for $G(V, E) \in training \cup testing$ **do**
  $SC^{\mathcal{A}} \leftarrow 0$
  $SC^{\mathcal{B}} \leftarrow 0$
  if $G(V, E) \in training$ **then** ▷ Seleccionamos los metagrafos correspondientes.
    if $G \in \mathcal{A}$ **then**
      $MG^{\mathcal{A}} \leftarrow$ Metagrafo de la clase $\mathcal{A}$ específico para G
      $MG^{\mathcal{B}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{B}$
    else
      $MG^{\mathcal{A}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{A}$
      $MG^{\mathcal{B}} \leftarrow$ Metagrafo de la clase $\mathcal{B}$ específico para G
    end if
  else
    $MG^{\mathcal{A}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{A}$
    $MG^{\mathcal{B}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{B}$
  end if
  for $e \in E$ **do**
    $SC_c^{\mathcal{A}} \leftarrow SC_c^{\mathcal{A}} + edgeBetweenness^{\mathcal{A}}(e)$ ▷ Sumamos el *edge betweenness* en vez del peso.
    $SC_c^{\mathcal{B}} \leftarrow SC_c^{\mathcal{B}} + edgeBetweenness^{\mathcal{B}}(e)$
  end for
  if $G(V, E) \in training$ **then**
    $predictoresEntrenamiento.append((SC^{\mathcal{A}}, SC^{\mathcal{B}}))$
  else
    $predictoresTesteo.append((SC^{\mathcal{A}}, SC^{\mathcal{B}}))$
  end if
end for
RL $\leftarrow regresionLogistica(predictoresEntrenamiento, [\mathcal{A}, \mathcal{A}, \mathcal{B}, ..., \mathcal{A}])$
Predicciones $\leftarrow RL.clase(predictoresTesteo)$

Únicamente será necesario actualizar la función de puntuación para que compute la intermediación en lugar de la proporción de veces que ocurren las aristas. El *edge betweenness* que usaremos será exacto y normalizado, ya que los grafos pueden diferir en número de aristas. Los nuevos resultados que obtenemos ahora son los mostrados en las Tablas 8 y 9.

Logistic Regression Classification (all edges, edge betweenness)											
Theta	All	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Adolescents	0.5671	0.5200	0.5164	0.5680	0.5782	0.6020	0.6459	0.5586	0.4955	0.5268	0.5076
Eyesclosed	0.5973	0.4886	0.5800	0.6227	0.5946	0.5843	0.6177	0.5575	0.5512	0.5446	0.5230
Male	0.6104	0.4939	0.5683	0.5741	0.5605	0.5916	0.5890	0.5921	0.5580	0.5618	0.5671
Children	0.4800	0.5443	0.5024	0.4729	0.5859	0.5400	0.4878	0.4184	0.4615	0.4557	0.4623

Tabla 8: AUC del modelo de regresión logística vía *edge betweenness*.

Logistic Regression Classification (exclusive edges, edge betweenness)											
Theta	All	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Adolescents	0.6144	0.5138	0.5444	0.6005	0.6099	0.6465	0.6747	0.5506	0.5473	0.5356	0.5231
Eyesclosed	0.5684	0.4972	0.5441	0.6054	0.5591	0.5584	0.5626	0.5276	0.5132	0.5880	0.4823
Male	0.6226	0.5146	0.6009	0.6220	0.5774	0.6377	0.5947	0.5920	0.5749	0.5734	0.5417
Children	0.4631	0.5392	0.5286	0.4933	0.5663	0.5561	0.5004	0.3984	0.4400	0.4847	0.4884

Tabla 9: AUC del modelo de regresión logística vía *edge betweenness* (con aristas exclusivas).

Si además queremos probar una solución híbrida, podemos ajustar el algoritmo para que tenga en cuenta ambos parámetros (intermediación de aristas y similitud con los metagrafos) de forma simultánea:

Algorithm 6 Regresión logística vía intermediación de las aristas y puntuación de similitud

$entrenamiento \leftarrow$ Grafos de entrenamiento
$testeo \leftarrow$ Grafos a clasificar
$predictoresEntrenamiento \leftarrow [\,]$
$predictoresTesteo \leftarrow [\,]$
$MG^{\mathcal{A}} \leftarrow$ Metagrafo de la clase $\mathcal{A}$
$MG^{\mathcal{B}} \leftarrow$ Metagrafo de la clase $\mathcal{B}$
for $G(V, E) \in training \cup testing$ **do**
  $SC^{\mathcal{A}} \leftarrow 0$
  $SC^{\mathcal{B}} \leftarrow 0$
  if $G(V, E) \in training$ **then** ▷ Seleccionamos los metagrafos correspondientes.
    if $G \in \mathcal{A}$ **then**
      $MG^{\mathcal{A}} \leftarrow$ Metagrafo de la clase $\mathcal{A}$ específico para G
      $MG^{\mathcal{B}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{B}$
    else
      $MG^{\mathcal{A}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{A}$
      $MG^{\mathcal{B}} \leftarrow$ Metagrafo de la clase $\mathcal{B}$ específico para G
    end if
  else
    $MG^{\mathcal{A}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{A}$
    $MG^{\mathcal{B}} \leftarrow$ Metagrafo genérico de la clase $\mathcal{B}$
  end if
  for $e \in E$ **do**
    $SC_c^{\mathcal{A}} \leftarrow SC_c^{\mathcal{A}} + edgeBetweenness^{\mathcal{A}}(e) * w^{\mathcal{A}}(e)$ ▷ *edge betweenness * similitud.*
    $SC_c^{\mathcal{B}} \leftarrow SC_c^{\mathcal{B}} + edgeBetweenness^{\mathcal{B}}(e) * w^{\mathcal{B}}(e)$
  end for
  if $G(V, E) \in training$ **then**
    $predictoresEntrenamiento.append((SC^{\mathcal{A}}, SC^{\mathcal{B}}))$
  else
    $predictoresTesteo.append((SC^{\mathcal{A}}, SC^{\mathcal{B}}))$
  end if
end for
RL $\leftarrow regresionLogistica(predictoresEntrenamiento, [\mathcal{A}, \mathcal{A}, \mathcal{B}, ..., \mathcal{A}])$
Predicciones $\leftarrow RL.clase(predictoresTesteo)$

De esta forma, los resultados obtenidos se reflejan en las Tablas 10 y 11.

Logistic Regression Classification (all edges, edge proportion + edge betweenness)											
Theta	All	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Adolescents	0.5454	0.5215	0.5170	0.5398	0.5829	0.5950	0.6316	0.5757	0.4986	0.5363	0.5063
Eyesclosed	0.6024	0.5484	0.5552	0.5796	0.5875	0.5922	0.6225	0.5756	0.5635	0.5438	0.5238
Male	0.6032	0.5444	0.5621	0.5628	0.5618	0.5729	0.5878	0.5932	0.5589	0.5649	0.5657
Children	0.4823	0.5255	0.5071	0.4886	0.5502	0.5110	0.4874	0.4502	0.4600	0.4580	0.4592

Tabla 10: AUC del modelo de regresión logística vía *edge betweenness* y puntuación de similitud.

Logistic Regression Classification (exclusive edges, edge proportion + edge betweenness)											
Theta	All	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Adolescents	0.5822	0.5026	0.5390	0.5517	0.5663	0.6348	0.6664	0.5507	0.5506	0.5418	0.5233
Eyesclosed	0.5648	0.5017	0.5410	0.5496	0.5346	0.5454	0.5607	0.5248	0.5112	0.5886	0.4824
Male	0.6134	0.5141	0.5580	0.5790	0.5765	0.6375	0.5967	0.5901	0.5745	0.5730	0.5415
Children	0.4482	0.5251	0.5149	0.4922	0.5275	0.5235	0.4969	0.4015	0.4435	0.4882	0.4920

Tabla 11: AUC del modelo de regresión logística vía *edge betweenness* y puntuación de similitud (con aristas exclusivas).

Comparando las anteriores Tablas, podemos resumir los mejores resultados de la siguiente forma:

Mejor AUC	**Adolescents**	**Eyesclosed**	**Male**	**Children**
Tabla 8	0.6459	0.6227	0.6104	0.5443
Tabla 9	0.6747	0.6054	0.6377	0.5561
Tabla 10	0.6316	0.6225	0.6032	0.5502
Tabla 11	0.6664	0.5886	0.6375	0.5275

Tabla 12: Mejores AUCs de los modelos de regresión logística con *edge betweenness*

Analizando la Tabla 12, podemos concluir que éstos modelos obtienen resultados muy similares y no hay una ventaja clara de ninguno de ellos frente a otro. Además, todos ellos obtienen el mejor rendimiento sobre el grupo *Adolescents* y el peor sobre el grupo *Children*.

4.8. Aproximación a la cercanía

Una vez hemos experimentado con las anteriores medidas de centralidad dirigidas a las aristas, demos un nuevo enfoque al problema. Partimos de un conjunto de grafos en los que todos los vértices son iguales, es decir, únicamente difieren en las aristas que los conectan. Sin embargo, podemos extraer atributos de los nodos que nos ayuden a mejorar nuestro clasificador. Es aquí donde entra la medida de cercanía.

La cercanía, o *closeness* en inglés, determina la distancia media de un nodo del grafo a todos los demás. Dejando a un lado los metagrafos con los que hemos trabajado hasta ahora, modificaremos nuestro algoritmo para que trate de clasificar los datos en función de este nuevo atributo.

Algorithm 7 Regresión logística vía cercanía

```
entrenamiento ← Grafos de entrenamiento
testeo ← Grafos a clasificar
predictoresEntrenamiento ← [ ]
predictoresTesteo ← [ ]
for G(V, E) ∈ training ∪ testing do
    C ← ⋃_{j=0}^{|V|} closeness(v_j)        ▷ Computamos la cercanía de todos los nodos de G.
    if G(V, E) ∈ training then
        predictoresEntrenamiento.append(C)
    else
        predictoresTesteo.append(C)
    end if
end for
RL ← regresionLogistica(predictoresEntrenamiento, [A, A, B, ..., A])
Predicciones ← RL.clase(predictoresTesteo)
```

En esta ocasión, en vez de entrenar el modelo con una lista de pares de la forma $[(SC_0^{\mathcal{A}}, SC_0^{\mathcal{B}}), ..., (SC_i^{\mathcal{A}}, SC_i^{\mathcal{B}})]$, nuestro nuevo input tendrá la forma $[(C_{0,0}, ..., C_{0,|V|}),$ $..., (C_{j,0}, ..., C_{j,|V|})]$. Tras realizar las pruebas con nuestro programa, obtenemos los siguientes resultados:

Logistic Regression (closeness)	
Adolescents	0.5268
Eyesclosed	0.5610
Male	0.5474
Children	0.4733

Tabla 13: AUC del modelo de regresión logística vía *closeness*.

Los valores obtenidos (Tabla 13) no son especialmente buenos, rondando el 0.5 (50 % de precisión). Claramente este método no funciona tan bien como otros, pero nos ayuda a entender qué clase de aproximaciones nos acercan más a nuestro objetivo.

4.9. Aproximación a los eigenvectores

Continuando el enfoque anterior sobre explorar métricas aplicables a los vértices en lugar de las aristas, podemos proponer una nueva idea: los eigenvectores.

Los eigenvectores o vectores propios miden la influencia de los nodos en el grafo asignando puntuaciones a los vértices en base a cómo de influyentes son sus vecinos. Analizando la distribución de estas puntuaciones en los grafos de nuestro *dataset*, quizás podamos extraer una forma de discriminarlos entre ambas clases más claramente.

Algorithm 8 Regresión logística vía eigenvectores

entrenamiento ← Grafos de entrenamiento
testeo ← Grafos a clasificar
predictoresEntrenamiento ← [ ]
predictoresTesteo ← [ ]
for $G(V, E) \in training \cup testing$ **do**
    $EV \leftarrow \bigcup_{j=0}^{|V|} eigenvector(v_j)$ ▷ Computamos los eigenvectores de todos los nodos de G.
    $EV \leftarrow normalizar(EV)$
    if $G(V, E) \in training$ **then**
        $predictoresEntrenamiento.append(C)$
    else
        $predictoresTesteo.append(C)$
    end if
end for
RL ← $regresionLogistica(predictoresEntrenamiento, [\mathcal{A}, \mathcal{A}, \mathcal{B}, ..., \mathcal{A}])$
Predicciones ← $RL.clase(predictoresTesteo)$

Siguiendo el Algoritmo 8, para cada grafo a clasificar calculamos los vectores propios de sus vértices [5]. Estos valores los podemos normalizar siguiendo la fórmula

$$EV_{normalizado} = \frac{max(EV)}{max(EV) - min(EV)}$$

para evitar de nuevo que la diferencia entre el número de conexiones de los grafos interfiera negativamente en la clasificación. Tras realizar las pruebas con nuestro programa, obtenemos los siguientes resultados (Tabla 14):

Logistic Regression (eigenvector)	
Adolescents	0.4866
Eyesclosed	0.5538
Male	0.5342
Children	0.6264

Logistic Regression (normalized eigenvector)	
Adolescents	0.4636
Eyesclosed	0.4745
Male	0.5399
Children	0.6028

Tabla 14: AUCs de los modelos de regresión logística vía eigenvectores.

Lamentablemente, para nuestro clasificador sigue siendo un desafío diferenciar entre los cerebros diagnosticados con *asd* y aquellos de la clase *td*. Únicamente se supera el 0.60 para la case *Children* (y de forma escasa), permaneciendo los demás valores en el rango (0.46, 0.54).

4.10. Redes neuronales artificiales como herramienta de clasificación

Puntuación por similitud, intermediación de las aristas, cercanía y eigenvectores son las medidas que hemos probado hasta ahora para clasificar las redes cerebrales. Las métricas anteriores nos permiten extraer información de los grafos, tanto de sus vértices como de sus aristas, con la que poder entrenar nuestro clasificador. Sin embargo, toda la experimentación realizada se ha llevada a cabo empleando el mismo tipo de modelo: la regresión logística.

[5] En el código estableceremos un máximo de 400 iteraciones.

A pesar de que la regresión logística suele ser una herramienta útil a la hora de clasificar variables dicotómicas, sería interesante comprobar la eficacia en la resolución del problema de otro modelo distinto. Para ello, haremos uso de las redes neuronales artificiales (ANN).

El diseño de la ANN es fundamental para optimizar su funcionamiento, adaptándola a nuestra tarea a resolver. Tras probar diversas configuraciones de forma manual, se ha optado por la estructura representada en la Tabla 15.

Capa	1	2	3	4	5	6	7
Neuronas	2 o 116	32	128	128	128	32	1
Función de activación	relu	relu	relu	relu	relu	relu	sigmoide

Tabla 15: Estructura de la ANN.

El tamaño de la capa 1 siempre será igual a la cantidad de información que nos aporte la medida de centralidad que estemos usando. En otras palabras, para las medidas centradas en las aristas, el tamaño de la primera capa será de 2 unidades, correspondiente a las puntuaciones $SC^{\mathcal{A}}$ y $SC^{\mathcal{B}}$ que emplearemos para clasificar cada grafo. Por otro lado, al usar medidas de centralidad enfocadas en los vértices, el tamaño de la capa será de 116 unidades, correspondiente al número total de nodos de cada grafo.

Las capas 2 y 6 constarán de 32 neuronas. Estas capas se usarán de puente entre las más pequeñas y más grandes de nuestra ANN, conformando una ampliación y reducción progresivas.

Las capas 3, 4 y 5 serán el core de nuestra red neuronal, ya que cada una de ellas constará de 128 neuronas. Este número se ha escogido por el balance que proporciona, siendo lo suficientemente alto como para que los pesos se configuren de una forma razonable (exista un aprendizaje), pero lo suficientemente bajo para que el modelo pueda entrenarse en un tiempo realista. Además. la función de activación de todas las capas nombradas hasta ahora será ReLU.

Finalmente, la última capa constará de una única unidad cuya función de activación será sigmoide. Puesto que la variable que buscamos predecir es binaria (el grafo pertenece o no pertenece a una clase), solo será necesaria una neurona que estime esta probabilidad.

En cuanto a características intrínsecas al comportamiento de la red neuronal, podemos destacar la importancia del optimizador y la función de pérdida. El optimizador es el mecanismo que ajustará los pesos de las unidades comparando la predicción con la función de pérdida. En nuestro caso emplearemos *adam*. Por otro lado, la función de pérdida se usa para encontrar errores o desviaciones en el proceso de aprendizaje. La función escogida será *binary cross entropy*.

Una vez definida la estructura y comportamiento de la ANN, debemos entrenarla con nuestros datos. Tres atributos clave decidirán cómo se llevará a cabo este entrenamiento: el tamaño del lote, el número de épocas y la fracción de validación. El tamaño del lote hace referencia al número de muestras por cada actualización del gradiente. El número de épocas será la cantidad de veces que se repite el entrenamiento con un lote para perfeccionar los pesos. La fracción de validación determinará el porcentaje de datos que serán

usados para validación y para testing.

El funcionamiento de nuestro nuevo programa será el siguiente:

Algorithm 9 Red Neuronal Artificial

$entrenamiento \leftarrow$ Grafos de entrenamiento
$testeo \leftarrow$ Grafos a clasificar
$predictoresEntrenamiento \leftarrow [\,]$
$predictoresTesteo \leftarrow [\,]$
for $G(V,E) \in training \cup testing$ **do**
  $M \leftarrow metrica(G_c)$ ▷ Computamos cualquier medida de centralidad de G_c.
  if $G(V,E) \in training$ **then**
    $predictoresEntrenamiento.append(C)$
  else
    $predictoresTesteo.append(C)$
  end if
end for
$ANN \leftarrow redNeuronal()$ ▷ Creamos la ANN y definimos sus características.
$ANN.anhadeCapa(unidades = |M_c|,\ activacion = relu)$
$ANN.anhadeCapa(unidades = 32,\ activacion = relu)$
$ANN.anhadeCapa(unidades = 128,\ activacion = relu)$
$ANN.anhadeCapa(unidades = 128,\ activacion = relu)$
$ANN.anhadeCapa(unidades = 128,\ activacion = relu)$
$ANN.anhadeCapa(unidades = 32,\ activacion = relu)$
$ANN.anhadeCapa(unidades = 1,\ activacion = sigmoide)$
$ANN.compilar(optimizador = adam,\ perdida = binary\_cross\_entropy)$
$ANN.entrenar(predictores = predictoresEntrenamiento,\ etiquetas = [\mathcal{A}, \mathcal{A}, \mathcal{B}, ..., \mathcal{A}],\ lote = 10,\ epocas = 50,\ validacion = 0{,}2)$
$\text{Predicciones} \leftarrow NN.clase(predictoresTesteo)$

Llegado el momento de realizar las pruebas, deberemos especificar qué caminos seguir y cuáles descartar. Dado que hasta ahora, parece que el rendimiento del clasificador es siempre mejor en las situaciones en las que $\theta = 0$ o $\theta = All$, sería ideal comprobar esos casos antes de iterar por todas las thetas una a una. También parece que excluir las aristas comunes a ambos metagrafos a la hora de calcular las puntuaciones no influye en gran medida en el resultados, ni de forma positiva ni negativa, por lo que realizaremos los experimentos teniendo en cuenta todas las aristas, al igual que en los primeros enfoques.

Tras ejecutar el código, se han obtenido los siguientes resultados:

- Clasificación de la red neuronal vía puntuación de similitud de los metagrafos (Tabla 16).

ANN (all edges, edge proportion)		
Theta	All	0
Adolescents	0.6100	0.6150
Eyesclosed	0.5474	0.5291
Male	0.5214	0.5448
Children	0.5212	0.5361

Tabla 16: AUC de la ANN via puntuaciones de similitud.

- Clasificación de la red neuronal vía *edge betweenness* (Tabla 17).

ANN (all edges, edge betweenness)		
Theta	All	0
Adolescents	0.5599	0.5053
Eyesclosed	0.5699	0.4992
Male	0.5443	0.5011
Children	0.4568	0.5122

ANN (all edges, edge proportion * edge betweenness)		
Theta	All	0
Adolescents	0.5494	0.5419
Eyesclosed	0.6048	0.5785
Male	0.5277	0.5201
Children	0.4898	0.4961

Tabla 17: AUCs de las ANN vía *edge betweenness*.

- Clasificación de la red neuronal vía *closeness* (Tabla 18).

ANN (closeness)	
Adolescents	0.5123
Eyesclosed	0.4964
Male	0.5073
Children	0.5004

Tabla 18: AUC de la ANN vía *closeness*.

- Clasificación de la red neuronal vía eigenvectores (Tabla 19).

ANN (eigenvector)	
Adolescents	0.4105
Eyesclosed	0.4850
Male	0.5635
Children	0.6432

ANN (normalized eigenvector)	
Adolescents	0.4773
Eyesclosed	0.5077
Male	0.5566
Children	0.5141

Tabla 19: AUCs de la ANN vía eigenvectores.

A primera vista podemos concluir que los resultados no son especialmente buenos. Por lo general, a nuestro nuevo modelo le resulta complicado clasificar correctamente los grafos empleando redes neuronales artificiales, al menos basándose en las medidas de centralidad anteriormente expuestas.

5. Conclusiones

Separar los datos según los grupos a los que corresponden es de vital importancia ya que distintas aproximaciones pueden funcionar de manera distinta dependiendo de numerosas causas, por ejemplo, diferencias biológicas presentes según el rango de edad de las muestras. Por ello, en la Tabla 20 representaremos el área bajo la curva ROC resultante tras la clasificación de muestras variando el método de clasificación así como la medida de centralidad empleada y separando los datos en los cuatro grupos del *dataset*.

Adolescents	Logistic Regression		ANN	
Theta	All	0	All	0
Metagraph Similarity	0.6540	0.6539	0.6100	0.6150
Edge Betweenness	0.5671	0.5200	0.5599	0.5053
Edge Betweenness * Metagraph Similarity	0.5454	0.5215	0.5494	0.5419
Closeness	0.5268		0.5123	
Eigenvectors	0.4866		0.4105	
Normalized Eigenvectors	0.4636		0.4773	

Eyesclosed	Logistic Regression		ANN	
Theta	All	0	All	0
Metagraph Similarity	0.5471	0.5901	0.5474	0.5291
Edge Betweenness	0.5973	0.4886	0.5699	0.4992
Edge Betweenness * Metagraph Similarity	0.6024	0.5484	0.6048	0.5785
Closeness	0.5610		0.4964	
Eigenvectors	0.5538		0.4850	
Normalized Eigenvectors	0.4745		0.5077	

Male	Logistic Regression		ANN	
Theta	All	0	All	0
Metagraph Similarity	0.5926	0.6510	0.5214	0.5448
Edge Betweenness	0.6104	0.4939	0.5443	0.5011
Edge Betweenness * Metagraph Similarity	0.6032	0.5444	0.5277	0.5201
Closeness	0.5474		0.5073	
Eigenvectors	0.5342		0.5635	
Normalized Eigenvectors	0.5399		0.5566	

Children	Logistic Regression		ANN	
Theta	All	0	All	0
Metagraph Similarity	0.6283	0.5863	0.5212	0.5361
Edge Betweenness	0.4800	0.5443	0.4568	0.5122
Edge Betweenness * Metagraph Similarity	0.4823	0.5255	0.4898	0.4961
Closeness	0.4733		0.5004	
Eigenvectors	0.6264		0.6432	
Normalized Eigenvectors	0.6028		0.5141	

Tabla 20: Comparación de resultados de la experimentación.

Al trabajar con el grupo *Adolescents*, está claro que el método de clasificación que mejor funciona es la regresión logística usando como variables predictoras las puntuaciones de similitud con los metagrafos (tanto para $\theta = All$ como para $\theta = 0$), seguido de las redes neuronales empleando las mismas herramientas.

En cuanto al grupo *Eyesclosed* es mucho más complicado decidir qué método funciona mejor. La regresión logística sobre puntuaciones de similitud ($\theta = 0$), *edge betweenness* ($\theta = All$) y la combinación *edge betweenness* * puntuaciones de similitud ($\theta = All$) junto

a las ANNs que usan ésta misma última medida (también con $\theta = All$), mantienen resultados muy parecidos, diferenciándose entre ellas en un factor menor de 0.015.

Con respecto al grupo *Male*, el grupo de mayor tamaño, la regresión logística vía puntuaciones de similitud con $\theta = 0$ parece conseguir resultados especialmente buenos en comparación con el resto de medidas.

El grupo *Children* parece ser el único que difiere un poco más en sus puntuaciones, consiguiendo la mejor empleando redes neuronales como método de clasificación y eigenvectores como medida de centralidad. Aunque el segundo mejor resultado vuelve a ser mediante la regresión logística y las puntuaciones de similitud ($\theta = All$) estando separado del mejor por 0.0149. A pesar de que esto puede deberse a diversos motivos, lo más probable es que el reducido número de datos de este grupo haya condicionado este resultado.

Resumiendo los mejores resultados en la Tabla 21:

Grupo	**Clasificador**	**Medida de centralidad**	θ	**AUC**
Adolescents	RL	Similitud a los metagrafos	All	0.6540
Eyesclosed	ANN	Similitud a los metagrafos * *edge betweenness*	All	0.6048
Male	RL	Similitud a los metagrafos	0	0.6510
Children	ANN	Eigenvectores	-	0.6432

Tabla 21: Mejores AUCs tras la experimentación completa.

Como conclusión, podemos indicar que las mejores formas de clasificar las redes cerebrales biológicas de cada grupo de nuestro conjunto de datos entre las investigadas en este trabajo son las anteriormente expuestas. Sin embargo, en caso de necesitar un clasificador global para todos los individuos, la solución óptima consistiría en elegir la regresión logística y las puntuaciones de similitud a los metagrafos.

6. Líneas futuras

En este trabajo hemos podido comprobar la eficacia de un conjunto de métodos de clasificación y medidas de centralidad de grafos para resolver el problema de la diferenciación entre cerebros con presencia del trastorno del espectro autista y cerebros neurotípicos.

Sin embargo, existen muchas formas de continuar esta investigación en busca de una aproximación que nos acerque más a los resultados deseados:

- Por un lado, podríamos explorar nuevos modelos de clasificación de variables dicotómicas más allá de la regresión logística y las redes neuronales convencionales. Por ejemplo, sería interesante realizar una aproximación al problema mediante redes neuronales gráficas, en inglés *graph neural networks* (GNNs), ya que éstas se especializan en grafos.
- Por otro lado, se podrían buscar otras medidas de centralidad que se adapten mejor a las características del problema. Dado que los mejores resultados de éste trabajo se han conseguido en la aproximación de los metagrafos, también sería posible continuar por ésta línea.
- Finalmente, otro camino a seguir sería realizar una experimentación con los métodos propuestos pero variando el conjunto de datos. Se podría tanto comprobar el comportamiento de estos modelos con un *dataset* de mayor tamaño, puesto que éste es muy limitado, como probar con representaciones cerebrales de conjuntos con otras anomalías neurológicas, como puede ser el alzhéimer, que presenten otro tipo de diferencias estructurales y/o temporales.

Referencias

[1] Colaboradores de los proyectos Wikimedia. (2003, Julio 14). *Neurociencia*. Wikipedia.org; Wikimedia Foundation, Inc. https://es.wikipedia.org/wiki/Neurociencia. Accedido el 20-06-2024.

[2] Sporns, O., Tononi, G., & Kötter, R. (2005). The Human Connectome: A Structural Description of the Human Brain. *PLoS Computational Biology*, 1(4), e42. https://doi.org/10.1371/journal.pcbi.0010042.

[3] Lanciano, T., Bonchi, F., & Gionis, A. (2020). Explainable Classification of Brain Networks via Contrast Subgraphs. *IRIS Research Product Catalog (Sapienza University of Rome)*. https://doi.org/10.1145/3394486.3403383.

[4] Faja, S., & Dawson, G. (2017). Autism Spectrum Disorder. *Child and Adolescent Psychopathology*, 745–782. https://doi.org/10.1002/9781394258932.ch22.

[5] Colaboradores de los proyectos Wikimedia. (2007, Noviembre 29). *Imagen por resonancia magnética*. Wikipedia.org; Wikimedia Foundation, Inc. https://es.wikipedia.org/wiki/Imagen_por_resonancia_magn%C3%A9tica. Accedido el 02-04-2024.

[6] Colaboradores de los proyectos Wikimedia. (2006, Noviembrer 24). *Imagen por resonancia magnética funcional*. Wikipedia.org; Wikimedia Foundation, Inc. https://es.wikipedia.org/wiki/Imagen_por_resonancia_magn%C3%A9tica_funcional. Accedido el 02-04-2024.

[7] Brandes, U. (2008). On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2), 136–145. https://doi.org/10.1016/j.socnet.2007.11.001

[8] Freeman, L. C. (1978). Centrality in Social Networks Conceptual Clarification. *Social Networks*, vol. 1, no. 3, Jan. 1978, pp. 215–239, https://doi.org/10.1016/0378-8733(78)90021-7.

[9] Zhang, J., & Luo, Y. (2017, Marzo 1) Degree Centrality, Betweenness Centrality, and Closeness Centrality in Social Network. *Atlantis Press*, https://www.atlantis-press.com/proceedings/msam-17/25874733.

[10] Colaboradores de los proyectos Wikimedia. (2021, March 8). Vector, valor y espacio propios. Wikipedia.org; Wikimedia Foundation, Inc. https://es.wikipedia.org/wiki/Vector,_valor_y_espacio_propios. Accedido el 22-04-2024.

[11] Negre, C. F. A., Morzan, U. N., Hendrickson, H. P., Pal, R., Lisi, G. P., Loria, J. P., Rivalta, I., Ho, J., & Batista, V. S. (2018). Eigenvector centrality for characterization of protein allosteric pathways. *Proceedings of the National Academy of Sciences*, 115(52), E12201–E12208. https://doi.org/10.1073/pnas.1810452115.

[12] Newman, M. The mathematics of networks. https://websites.umich.edu/~mejn/papers/palgrave.pdf

[13] Stock, J. H., & Internet Archive. (2015). Introduction to Econometrics. *Internet Archive*, Boston : Pearson. https://archive.org/details/introductiontoec0000stoc_z0a9/page/800/mode/2up.

[14] SStoltzfus, J. C. (2011). Logistic Regression: A Brief Primer. *Academic Emergency Medicine*, vol. 18, no. 10, Oct. 2011, pp. 1099–1104, https://doi.org/10.1111/j.1553-2712.2011.01185.x.

[15] Dongare, A., Kharde, R., & Kachare, A. (2008). Introduction to Artificial Neural Network. *Certified International Journal of Engineering and Innovative Technology (IJEIT)*, 9001(1), 2277–3754. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=04d0b6952a4f0c7203577afc9476c2fcab2cba06.

[16] Bre, F., et al. (2018). Prediction of Wind Pressure Coefficients on Building Surfaces Using Artificial Neural Networks. *Energy and Buildings*, vol. 158 pp. 1429–1441. https://doi.org/10.1016/j.enbuild.2017.11.045.

[17] Zou, J., Han, Y., & So, S. (2008). Overview of Artificial Neural Networks.*Methods in Molecular Biology*^{TM}, 458, 14–22. https://doi.org/10.1007/978-1-60327-101-1_2.

[18] Sharma, S. (2017, Septiembre 6). Activation Functions in Neural Networks. *Medium, Towards Data Science*. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[19] Grover, P. (2022, December 17). Applications of Different Parts of an ROC Curve. *Medium*. https://medium.com/towards-data-science/applications-of-different-parts-of-an-roc-curve-b534b1aafb68.

[20] Brownlee, J. (2018, August 30). How to Use ROC Curves and Precision-Recall Curves for Classification in Python. *Machine Learning Mastery*. https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/.

Anexo

El código empleado en el proyecto, así como el *dataset* al completo, se encuentra en el repositorio de GitHub https://github.com/carlos-vf/Brain-Network-Classification.git.