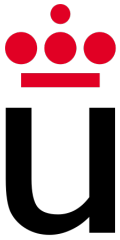


Universidad
Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
INFORMÁTICA

APRENDIZAJE AUTOMÁTICO EN CLASIFICACIÓN DE GRAFOS MOLECULARES

GRADO EN INGENIERÍA DEL SOFTWARE
TRABAJO FIN DE GRADO

Autor: Markos Aguirre Elorza
Director: Iván Ramírez Díaz

Curso Académico 2022/2023

Capítulo 1

Resumen

Resumen en Español

En este trabajo se aborda el problema de la clasificación de moléculas haciendo uso, en concreto, de técnicas de aprendizaje automático. Como objetivo principal, se plantea la implementación de distintos modelos de representación y clasificación para su posterior evaluación en términos de rendimiento.

En el capítulo 2 de introducción se va a explicar la necesidad para tratar este problema, la revolución que se ha llevado a cabo en estos últimos años gracias al aprendizaje automático y se detallará sobre qué cuestiones particulares versará este trabajo.

En el tercer capítulo, se detallarán los objetivos de este trabajo. Se describirá resumidamente el problema, se expondrá la metodología empleada para lograr el objetivo y se comentarán las alternativas que ha habido a este enfoque.

Los siguientes cuatro capítulos siguen esta lógica. En los capítulos 4 y 5, se expone el estado del arte. Concretamente, en la cuarta sección se recoge cómo se representan las moléculas de forma que se puedan procesar y en la quinta qué modelos existen para clasificar esos datos ya preprocesados. Siguiendo el mismo esquema, en los capítulos 6 y 7 se explica el estudio comparativo llevado a cabo para la ocasión. En el capítulo seis se detallan los datos escogidos para el estudio, mientras que en el séptimo se valora la aplicación de los modelos de aprendizaje escogidos.

Concretamente, en el capítulo 4, se van a explicar las distintas formas que existen para representar moléculas, haciendo especial hincapié en las maneras convenientes para el clasificado automático. No es una cuestión trivial, ya que los modelos de aprendizaje automático normalmente necesitan trabajar con información vectorizada y esto obliga a que se haga un preprocesado de los datos cuidadoso.

En el quinto capítulo, se discutirán los distintos modelos de aprendizaje automático que se utilizan en esta tarea. Se expondrá el funcionamiento de los modelos más exitosos en este problema.

En el siguiente capítulo, se van a presentar los conjuntos de datos que se utilizarán en el estudio comparativo. Se explicará para qué se utilizan en escenarios

reales, qué propiedades tienen, si existe desbalanceo de datos o no, etc., y también se detallará qué descriptores moleculares se disponen en cada *dataset* para hacer el clasificado.

Por otro lado, ya en el capítulo 7, se hará un estudio comparativo entre cinco distintos modelos de aprendizaje automático introducidos en el capítulo 5 y con los cuatro distintos conjuntos de datos del capítulo 6. Se verá la importancia de una correcta implementación de los métodos escogidos y se valorarán los resultados obtenidos.

Para finalizar, se dedicarán unas páginas para exponer las conclusiones obtenidas. Principalmente, sobre el rendimiento y las limitaciones de los modelos y los descriptores tratados a lo largo del trabajo.

Summary in English

This work addresses the problem of molecule classification using, specifically, machine learning techniques. The main objective is the implementation of various representation and classification models for subsequent evaluation in terms of performance.

In Chapter 2, the introduction, we will explain the necessity of addressing this problem, discuss the revolution that has occurred in recent years due to machine learning, and detail the specific issues that this work covers.

In the third chapter, the objectives of this work will be detailed. The problem will be briefly described, the methodology used to achieve the goal will be presented, and the alternatives to this approach will be discussed.

The following four chapters follow this logic. Chapters 4 and 5 present the state of the art. Specifically, in the fourth section, how molecules are represented so they can be processed is collected, and in the fifth, which models exist to classify this preprocessed data. Following the same scheme, in chapters 6 and 7, the comparative study conducted for the occasion is explained. Chapter six details the data chosen for the study, while chapter seven assesses the application of the selected learning models.

In chapter 4, different ways to represent molecules will be explained, with a special emphasis on the convenient ways for automatic classification. It is not a trivial matter, as machine learning models usually need to work with vectorized information, which requires careful preprocessing of the data.

In the fifth chapter, the various machine learning models used in this task will be discussed. The operation of the most successful models in this problem will be exposed.

In the following chapter, the data sets that will be used in the comparative study will be presented. It will be explained what they are used for in real scenarios, what properties they have, whether there is data imbalance or not, etc., and also which molecular descriptors are available in each dataset for classification.

On the other hand, in chapter 7, a comparative study will be carried out between five different machine learning models introduced in chapter 5 and with the

four different data sets from chapter 6. The importance of correct implementation of the chosen methods will be seen and the results obtained will be evaluated.

Finally, a few pages will be dedicated to presenting the conclusions obtained. Mainly, on the performance and limitations of the models and descriptors dealt with throughout the work.

Palabras Clave: Clasificación molecular, aprendizaje automático, descriptor molecular, redes convolucionales sobre grafos

Keywords: Molecular classification, machine learning, molecular descriptor, graph convolutional networks

Índice general

Índice de figuras

Índice de tablas

Capítulo 2

Introducción

El clasificado de moléculas ha sido una tarea primordial en la química. No solo se ha limitado a cuestiones taxonómicas, sino que ha sido de gran utilidad para detectar enfermedades, distinguir fármacos potencialmente beneficiosos, detección de sustancias tóxicas, etc.

En los últimos años, con el auge de los modelos de aprendizaje automático, se ha computarizado esta tarea ahorrando muchos recursos y posibilitando vías de investigación que de otra forma no serían posibles. La combinación de la inteligencia artificial y el clasificado de moléculas ha obtenido muy buenos resultados [Mar22] [WH21].

En este trabajo, se van a estudiar cuatro conjuntos de datos que se han utilizado para lo siguiente:

1. Distinguir enzimas de moléculas que no lo son.
2. Detectar sustancias tóxicas.
3. Clasificar moléculas que interactúan con una bacteria determinada.
4. Detectar agentes activos en procesos de cáncer de pulmón.

Estos *datasets* se han escogido por su popularidad en tareas de clasificación y por su relevancia en escenarios reales.

Para poder hacer esta clasificación, primero hay que representar correctamente las moléculas. Estas formas de representación llamadas descriptores moleculares surgen de la necesidad de mostrar información relevante de la molécula en un formato que sea manejable. Por necesidad, se hará hincapié en los pormenores de los formatos que, en especial, sean los idóneos para modelos de aprendizaje automático. Después, se explicarán los modelos que sean adecuados para esta tarea.

Una vez expuesto el fundamento teórico de la clasificación molecular, se hará un estudio comparativo de cinco implementaciones distintas de aprendizaje automático, de forma que se puedan extraer las características que hacen a unos métodos mejores que a otros.

Capítulo 3

Objetivos

Cabe recalcar que el objetivo de este estudio es valorar el rendimiento de distintos modelos de aprendizaje automático, con diferentes conjuntos de datos y descriptores, en la clasificación de moléculas.

Las alternativas a estos métodos han ido cambiando a lo largo de la historia. Por ejemplo, tal y como se describe en este artículo del año 1999 [Gol+99], para clasificar la leucemia se recurría a especialistas y se hacía a través de la morfología del tumor. Después, los investigadores del artículo, propusieron un método automático de clasificado molecular basado en información citoquímica celular. Sin embargo, estos enfoques presentan problemas de subjetividad, sensibilidad a los datos de entrada y son costosas en tiempo y recursos. Por eso, es interesante valorar otras estrategias como el uso de la inteligencia artificial.

Para valorar el desempeño de los modelos de aprendizaje automático, se ha realizado un estudio comparativo con validación cruzada aleatoria. En total, se van a hacer 10 iteraciones independientes de 15 clasificaciones diferentes, midiendo parámetros como la desviación típica, precisión y tiempo de ejecución. De esta forma, se podrán extraer conclusiones válidas de los distintos modelos, descriptores moleculares y conjuntos de datos.

Capítulo 4

Formas de representación de moléculas

En este capítulo se va a hablar de las distintas formas que existen para representar moléculas. La manera más adecuada de representar la información molecular dependerá de la situación, sobre todo, de las características que se quieran reflejar. A las formas de representación de moléculas se las llamará **descriptores moleculares** y será la primera cuestión que se trate. Más tarde, se hablará sobre el descriptor molecular más popular para la clasificación automática de moléculas, el **grafo molecular**. Este descriptor molecular es un concepto muy amplio porque utiliza la idea de grafo, que no es más que un conjunto de objetos que permite representar relaciones binarias. Por este motivo, muchos descriptores llevarán esta idea implícitamente, como es en el caso del último punto a tratar en esta sección: los **SMILES**. En este trabajo también se va a utilizar otro descriptor: la imagen de persistencia. Esta será tratada en la sección 5.3.1 del capítulo 4 porque es un descriptor molecular muy dependiente del modelo de aprendizaje automático WKPI.

4.1. Descriptores moleculares

Como se ha dicho en la apertura de este capítulo, en líneas generales, un descriptor molecular es una forma de representar una molécula. Esto no es algo trivial, ya que el concepto de la estructura molecular es especialmente rico y complejo debido a la gran cantidad de información que aguarda una molécula, nótese que toda la información de un ser vivo se almacena en una molécula de ADN. Cada forma de representar una molécula es en realidad una forma diferente de verla, es decir, cada representación captura una parte de realidad de la molécula.

Una definición más formal del **decriptor molecular** la dan los científicos R. Todeschini y V. Consonni: ”un descriptor molecular es el resultado final de un procedimiento lógico y matemático que transforma información química codificada en una representación simbólica, en un número útil o en el resultado de un

experimento estandarizado".

Una línea de investigación muy viva es la búsqueda de distintos descriptores moleculares que sean capaces de capturar nuevos aspectos de sus estructuras. Como el químico M. Randic decía: "No hay restricciones en el diseño de invariantes estructurales, el factor limitante es la imaginación de uno mismo"[Ran96].

Los criterios que todo descriptor molecular debe seguir son:

- Que sean invariantes respecto al etiquetado y numeración de los átomos
- Invariantes a las rotaciones
- Tener una definición algorítmicamente computable que no sea ambigua
- Valores en un rango numérico apropiado (en las ocasiones oportunas)

Otros de los aspectos deseables para un buen descriptor molecular son:

- Una interpretación estructural
- Una buena correlación con al menos una propiedad
- No tener correlaciones triviales con otros descriptores moleculares
- No incluir en la definición propiedades experimentales
- No restringirse a un conjunto demasiado pequeño de moléculas
- Tener capacidad de discriminar isómeros
- Que se pueda decodificar de forma inversa

La idea general que se debe tener en mente para clasificar moléculas mediante descriptores moleculares es que de la misma manera que diferentes estructuras moleculares tienen distintas propiedades químicas y las estructuras moleculares similares tienen propiedades parecidas, las moléculas que se representen con descriptores parecidos tendrán propiedades similares y las moléculas con descriptores disímiles tendrán propiedades distintas.

Se pueden encontrar muchos descriptores creados con técnicas muy distintas que recojan información muy diferente como la topoestructural, topográfica, topoquímica, mecanicocúantica etc. En este artículo [WWK08] se describen cinco diferentes, sin embargo, en la siguiente sección se tratará los que mejores resultados han dado en tareas de clasificación automática: el grafo molecular.

4.2. Grafo molecular

En esta sección se tratará el grafo molecular, el descriptor más popular. Se comenzará los aspectos fundamentales del grafo molecular y, después, se tratarán los aspectos para la vectorización del grafo molecular. Estas cuestiones de vectorización son específicas de este descriptor y es necesario tratarlas para que se puedan clasificar usando técnicas de aprendizaje automático.

4.2.1. Aspectos fundamentales del grafo molecular

En primer lugar, se definirá un grafo, después un grafo molecular y, finalmente, se comentarán las características más relevantes del descriptor molecular.

Sea el grafo $G = (V, E)$ con los conjuntos de los vértices V y el de las aristas E. El **grafo molecular** es el grafo cuyos vértices representan átomos y cuyas aristas reflejan la unión entre dos átomos. De forma general, se trabaja sobre grafos simples, es decir, donde como mucho hay un enlace entre dos nodos, no hay enlaces entre un nodo y sí mismo y todas las aristas son no dirigidas. Además, los grafos que se utilicen serán conexos y podrán contener ciclos (muy típico en los bencenos, p. ej.).

Existen cuatro características de un grafo que son potencialmente interesantes: los nodos, las aristas, el contexto global y la conectividad. Dependiendo de la situación, será conveniente utilizar una clase de grafo que se adapte mejor a las características que se quiera reflejar, por ejemplo, utilizando un grafo etiquetado donde se añada un peso a las aristas dependiendo del tipo de enlace que represente.

A la hora de trabajar con este tipo de datos se va a necesitar unos conceptos que se definen más abajo.

El **grado** de un nodo viene determinado por el número de vecinos que tiene:

$$grad(u) = \sum_{v \in V} A[u, v]$$

donde A es una función que toma el valor de 1 sí u y v son adyacentes y 0 en caso contrario. El grado de un átomo puede ser interesante para medir la estabilidad y reactividad. Por ejemplo, un carbono con cuatro enlaces (tetravalente) es más estable que un carbono con tres enlaces (trivalente).

Otra medida importante es la **centralidad del autovector** e_u (*eigenvector centrality*). A diferencia de la medida de grado, que solo cuenta el número de conexiones o nodos adyacentes a un nodo específico, la centralidad de autovector también tiene en cuenta la importancia o relevancia de estos nodos adyacentes. Definiendo e_u recurrentemente:

$$e_u = \frac{1}{\lambda} \sum_{v \in V} A[u, v] e_v \quad \forall u \in V$$

donde λ es una constante. En notación vectorial se tiene $\lambda e = Ae$, es decir, λ es la medida de centralidad y se corresponde con el autovalor de la matriz adyacente. Se puede ver como la probabilidad de que un nodo sea visitado en un recorrido aleatorio infinito sobre el grafo. Esto se suele calcular mediante el método de las potencias, donde el vector resultante dirá el número de veces que se ha visitado un nodo en un camino aleatorio de longitud infinita.

4.2.2. Vectorización del grafo molecular

Si se quieren clasificar moléculas con técnicas de aprendizaje automático (exceptuando las redes neuronales sobre grafos), las moléculas deben estar representadas en un formato que el ordenador pueda manejar, como puede ser un vector, una matriz o una imagen. El primer paso es crear un descriptor molecular adecuado que recoja la información de la molécula. El segundo paso consiste en vectorizar este descriptor de forma que el ordenador pueda operar con él. De la misma manera que hay muchos descriptores para una sola molécula, existen multitud de distintas vectorizaciones para un solo descriptor. En las siguientes líneas se comentarán cuáles son y qué ventajas o desventajas ofrecen.

Antes de tratar de vectorizar la totalidad del grafo, cabe destacar que es muy habitual representar los nodos (átomos) junto a un *feature vector* o **vector de características** o propiedades en español, donde se recoge información como la masa atómica, carga eléctrica, etc. Estos vectores se representan con una matriz $X \in \mathbb{R}^{|V| \times m}$. Es decir, por cada átomo existiría un vector que recogería m características (en forma numérica normalmente) de ese átomo. Por ejemplo, en esta matriz

$$\begin{pmatrix} 1 & +1 \\ 1 & +1 \\ 8 & -2 \end{pmatrix}$$

se muestra el número atómico y la carga eléctrica para cada átomo de la molécula de agua. En este caso, los átomos de hidrógeno tienen un número atómico de 1 y una carga eléctrica de +1, mientras que el átomo de oxígeno tiene un número atómico de 8 y una carga eléctrica de -2.

Una forma vectorizada de representar los grafos (en su totalidad) es la matriz de adyacencia. La **matriz de adyacencia** A para un grafo $G(V, E)$, donde $V = \{v_1, ..., v_n\}$ es el conjunto de vértices, se define como:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \text{ donde } a_{ij} = \begin{cases} 1 \text{ si } v_i \text{ y } v_j \text{ son adyacentes} \\ 0 \text{ en caso contrario} \end{cases} \text{ con } i, j \in \{1, ..., n\}$$

La matriz será simétrica siempre y cuando las aristas sean no dirigidas, en caso contrario, $a_{ij} \neq a_{ji}$ y tomarán un valor que se defina para la ocasión. Si los enlaces son ponderados, a_{ij} tomará otro valor distinto a 0 o 1, por ejemplo, para indicar la fuerza de un enlace se podría definir $a_{ij} \in [0, 1] \in \mathbb{R}$.

Esta forma de representar grafos podría resultar beneficiosa, ya que los modelos de aprendizaje automático normalmente parten de un input en forma vectorial o matricial.

Sin embargo, la matriz de adyacencia quedaría dispersa (que es muy ineficiente en cuanto a espacio en memoria) en grafos muy grandes o con muchos enlaces. Tampoco se comportarían bien en matrices no tan grandes porque la conectividad

del grafo no es invariante ante la permutación, es decir, existen distintas matrices para representar el mismo grafo con distintos resultados. En la Figura 4.1, se puede observar un grafo muy simple de cuatro nodos ($n = 4$). Como se pueden permutar las columnas y filas de su matriz de adyacencia (sin alterar el grafo que se representa) el número de matrices viene dado por $n!$, es decir, 24 matrices de adyacencia para un solo grafo de 4 nodos. En la foto obtenida de [God18] se pueden observar las distintas matrices:

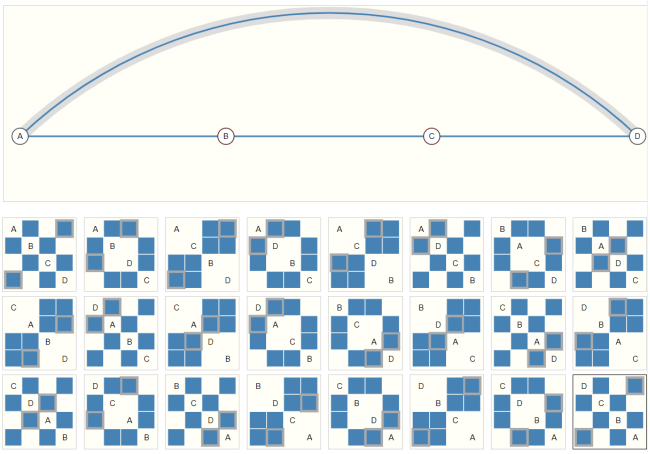


Figura 4.1: Distintas matrices de adyacencia

Otras de las matrices más relevantes son la **matriz de grados** D que tiene en la diagonal el grado de cada nodo (0 en el resto de celdas) y la **matriz Laplaciana** L que se define como $L = D - A$.

Por supuesto, existen más formas de representar grafos de forma vectorial e incluso existen modelos de aprendizaje automático que operan con grafos no vectorizados tal y como se explica en esta sección 5.1.

4.3. SMILES

En primer lugar, en la sección de aspectos fundamentales, se va a definir el descriptor molecular SMILES (*Simplified Molecular-Input Line-Entry System* y, después, se discutirán sus características. Por último, se describirá un algoritmo que genera este descriptor molecular.

4.3.1. Aspectos fundamentales

Los **SMILES** (*Simplified Molecular-Input Line-Entry System*) son unos descriptores moleculares que se representan mediante cadenas de texto. Fueron introducidas por David Weininger en 1988 [Wei88]. La idea general es recorrer los grafos de una manera representativa escribiendo los símbolos atómicos en una cadena de texto. Es una forma similar a la de las fórmulas químicas (la fórmula

química del agua es H_2O, por ejemplo) que son la representación de los elementos que forman un compuesto y la proporción en que estos se encuentran. Sin embargo, los SMILES recogen mejor varias propiedades de las moléculas.

La versión canónica de SMILES permite que una molécula quede inequívocamente y biyectivamente representada por un *string*. Para referirnos a los átomos, se utilizará su símbolo atómico (B, C, N, O, P, S, F, Cl, Br, e I) y los enlaces pueden ser omitidos (si son covalentes, simples o aromáticos), dibujados con = sin son dobles o con # si son triples. Las ramas se escriben entre paréntesis y las estructuras cíclicas son linealizadas rompiendo uno de los enlaces simples o aromáticos. Sin embargo, existen otras muchas excepciones y consideraciones que se van a omitir por no ser de especial interés en este trabajo.

Lo que sí es importante mencionar es que los SMILES comprenden la quiralidad, una propiedad de asimetría que se la conoce también como simetría de espejo. Las @ van a representar centros de quiralidad tetraédrica cuando están en una orientación antihoraria y @@ cuando están en orientación horaria.

Una vez mencionadas las reglas generales, se va a explicar cómo se generan inequívocamente los SMILES canónicos. La forma de obtener estos *strings* no es única, existen muchos softwares comerciales que las crean. Sin embargo, se tratará el algoritmo CANGEN aunque no siempre sea capaz de representar las moléculas de forma biyectiva, en concreto, porque existen compuestos simples que se traducen en varias cadenas de texto [NGL05]. A pesar de esto, es un algoritmo ampliamente citado en la investigación.

4.3.2. Algoritmo CANGEN

El algoritmo CANGEN, descrito en este artículo [WWW89], consta de dos partes: el CANON que etiqueta cada átomo inequívocamente basándose en su estructura topológica y el GENES que genera el SMILE partiendo de los átomos etiquetados y las reglas anteriormente mencionadas. Cabe destacar, que el algoritmo parte del descriptor molecular anteriormente visto, el grafo molecular. Sin embargo, el grafo solo se utiliza para saber cómo están unidos los átomos entre sí dentro de la molécula.

En las siguientes líneas se explicará cómo funciona el algoritmo CANON. Se genera un invariante por cada átomo. Un invariante es una propiedad de una molécula que no cambia, independientemente de la forma en que se represente la estructura molecular. Se crean según la siguiente información:

1. Número de conexiones (sin contar con átomos de hidrógeno)
2. Número de enlaces (sin contar con átomos de hidrógeno)
3. Número atómico
4. Signo de carga
5. Carga absoluta

6. Número de hidrógenos anexados

Cabe destacar que se puede añadir más información relevante. Por ejemplo, en la molécula del pentano, donde los vértices negros son átomos de carbono y los blancos átomos de hidrógeno: Los invariantes para los átomos carbono metilo

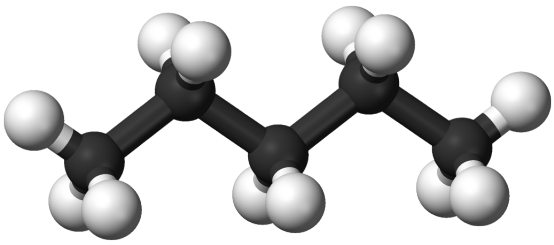


Figura 4.2: Pentano

(los carbonos con 3 hidrógenos que están en los extremos) y carbono metileno (los carbonos con 2 hidrógenos y 2 carbonos de en medio) en el pentano son los siguientes:

Átomo	Invariante
Metilo	116003
Metileno	226002

Como se puede observar, el invariante simplemente se calcula poniendo los números de la enumeración anterior en orden. Concretamente, observando el invariante para el átomo Metilo, tenemos que solo está unido a 1 átomo que no sea hidrógeno, como ese enlace es simple, en el siguiente lugar se pone otro 1, el número atómico del carbono es 6, la carga es neutra por lo que se tiene un 0, la carga absoluta también es 0 y el número de hidrógenos anexados es 3. En total, el pentano quedaría así:

116003-226002-226002-226002-116003

Que es equivalente a:

1-2-2-2-1

Para comprender propiedades simétricas, los conjuntos de invariantes, se pueden ver en términos de las sumas de los invariantes de los átomos con una distancia, dos distancias, etc. El test de simetría es si "las sumas conectivas extendidas" son diferentes.

Para el ejemplo del pentano, asumiendo que las vecindades están a una distancia, operando una función (usualmente la suma) sobre los vecinos de cada átomo. Es decir, asignando a cada átomo la suma de la invarianza atómica de sus vecinos, se obtiene lo siguiente:

2-3-4-3-2

O equivalentemente,

1-2-3-2-1

Este proceso se repite hasta que no hay más cambios. En este caso, no se puede ir más lejos porque las siguientes sumas darían lugar a 2-4-4-4-2 que es equivalente a 1-2-2-2-1 (vector del principio).

El problema de este método es que es ambiguo, puede haber coincidencias de suma de enteros que lleven a falsas ilusiones de simetría. Para ello, se utilizan productos de primos en vez de sumas. Sean dos átomos con tres átomos adyacentes con estos valores 1, 4, 4 y 2, 2, 5 las sumas darían el mismo valor, 9, y podrían dar una falsa impresión de simetría. Ahora, si se asigna un número primo por cada entero 2, 3, 3 y 5, 5, 7; y se multiplican, dan como resultado 2*3*3=18 y 5*5*7=175 que, efectivamente, indican que no hay simetría.

Si se encuentran nodos simétricos se rompen enlaces para poder hacer una representación inequívoca con el algoritmo siguiente. Como se ha comentado, se repite este paso hasta que no haya más variaciones.

El paso anterior deja todos los átomos de la molécula inequívocamente identificados. El algoritmo GENES comenzará siempre con el de menor valor. Las decisiones de ramificación se basan en seguir la rama hasta que muera y elegir el siguiente conforme al que menor valor tenga. Conforme se vayan recorriendo los átomos se irá escribiendo el*string* del SMILE.

El algoritmo CANON para una estructura de N átomos tiene una complejidad temporal $O(N^2 log(N))$. Esta se puede considerar elevada si el número de átomos de la molécula fuera grande, pero no suele ser el caso. Por tanto, la complejidad de este algoritmo no debería ser problemática.

Esta técnica podría resultar interesante si se quisieran clasificar las moléculas mediante técnicas de procesamiento de lenguaje natural (NLP en inglés). Como se puede ver en este trabajo [JLC16], se pueden conseguir resultados muy prometedores.

Capítulo 5

Métodos de aprendizaje automático para clasificación de moléculas

El aprendizaje automático es una rama de la inteligencia artificial que elabora modelos que son capaces de aprender a partir de datos, mejorar con la experiencia y tomar decisiones basadas en lo anterior. Se distingue de otros enfoques en que esta no sigue instrucciones explícitas predefinidas.

Este método posibilita resolver problemas complejos que no son abordables desde un punto de vista determinista. Se destacan la clasificación de imágenes, reconocimiento de voz, predicción del mercado de valores, etc. En este trabajo se evaluará si también son efectivos en clasificación molecular.

Se pueden distinguir dos familias de modelos de aprendizaje automático para el clasificado de moléculas: los que parten sobre grafos y los que se valen de información vectorizada (vectores, imágenes, etc.). La sección 4.1. es introductoria y explicará qué distintos tipos de tareas de aprendizaje automático se hacen sobre grados. En la sección 4.2. se tratan los modelos sobre grafos y en la sección 4.3. se versa sobre una técnica que parte del descriptor molecular grafo, lo concierte en otro descriptor llamado imagen de persistencia y aplica un modelo sobre esta forma vectorizada.

5.1. Tipos de tareas de aprendizaje automático en grafos

Antes de detallar el funcionamiento de los modelos concretos, se explica qué tipos de problemas se pueden afrontar con el aprendizaje automático en grafos. En primer lugar, conviene mencionar que el aprendizaje automático con grafos es algo diferente al que se hace con imágenes, por ejemplo. Una de las diferencias es que las categorías habituales de aprendizaje supervisado y no supervisado no son necesariamente las más convenientes cuando se trata de grafos. Esto es debido a que a menudo se difuminan los límites entre las dos categorías tradicionales.

De forma general, existen cuatro tipos de problemas con grafos (se descri-

ben en [Ham20]) que se relacionan con las cuatro características mencionadas anteriormente (los nodos, las aristas, la conectividad y el contexto global):

- **Clasificación de nodos**: Por ejemplo, averiguar qué tipo de átomo (carbono, hidrógeno, etc.) es un nodo determinado. El objetivo es predecir la etiqueta y_u (que podría ser un tipo, categoría o atributo) asociada con todos los nodos $u \in V$, cuando solo se dan las etiquetas verdaderas en un conjunto de entrenamiento $V_e \subsetneq V$. La clasificación de nodos es quizás la tarea de aprendizaje automático más popular en datos en forma de grafos, aunque no tanto en grafos moleculares.

  A primera vista, la clasificación de nodos parece ser una variación sencilla de la clasificación supervisada estándar, pero hay diferencias importantes. La más importante es que los nodos en un gráfico no están independiente e idénticamente distribuidos (i.i.d.). Por lo general, cuando se construyen modelos de aprendizaje automático supervisados, se asume que cada punto de datos es estadísticamente independiente de todos los demás puntos de datos; de lo contrario, se podría necesitar modelar las dependencias entre todos los puntos de entrada. De la misma forma, se supone que los puntos de datos se distribuyen de manera idéntica; de lo contrario, no se tendría forma de garantizar que el modelo se generalizase a nuevos puntos de datos. La clasificación de nodos rompe completamente esta suposición i.i.d., ya que se modela un conjunto interconectado de nodos. De hecho, se explota esta particularidad con mucho éxito. Concretamente con estas tres estrategias:

  - Haciendo uso de la homofilia que es la tendencia de los nodos a compartir características con sus adyacentes.
  - Haciendo uso de la equivalencia estructural que es la idea de que los nodos con estructuras de vecindario similares tendrán etiquetas parecidas.
  - Haciendo uso de la heterofilia que supone que los nodos estarán conectados preferentemente a nodos con etiquetas distintas.

  ¿Se podría considerar este problema como no supervisado? La verdad es que siempre se va a tener acceso a cierta información de los nodos, aunque pertenezcan al conjunto de entrenamiento y su etiqueta esté oculta, porque su lugar en el grafo y, por ende, su información estructural estará disponible. Por ello, se utiliza el término de semisupervisado.

- **Predicción de enlace**: Se parte de un conjunto de nodos V y un conjunto de entrenamiento de aristas $E_e \subsetneq E$. Nuestro objetivo es utilizar esta información parcial para inferir las aristas que faltan $E \setminus E_e$. La complejidad de esta tarea depende sobre todo del tamaño del grafo, con datos pequeños es viable encontrar una heurística para determinar cuántos vecinos comparten dos nodos, a medida que el grafo aumenta, esta búsqueda se complica.

Al igual que la clasificación de nodos, la predicción de relaciones difumina los límites de las categorías tradicionales de aprendizaje automático, a menudo denominándose tanto supervisada como no supervisada, y requiere sesgos inductivos específicos para el dominio del gráfico. Además, al igual que la clasificación de nodos, hay muchas variantes de predicción de relaciones, incluidas configuraciones donde las predicciones se realizan sobre un gráfico único, así como configuraciones donde las relaciones deben predecirse en múltiples gráficos disjuntos (grupos de moléculas de cierto tipo p. ej.).

- **Agrupamiento y detección de comunidades**: Tanto la clasificación de nodos como la predicción de relaciones requieren inferir información faltante sobre datos de gráficos, y en muchos aspectos, esas dos tareas son los análogos de gráficos al aprendizaje supervisado. La detección de comunidades, por otro lado, es el análogo de los grafos al agrupamiento no supervisado.

  Aunque quizás tengo más aplicaciones en grafos como redes sociales, donde, por ejemplo, se quisiera identificar los grupos de amigos que se conocen de la escuela. En grafos moleculares se podría querer identificar que átomos dentro de una molécula tienen la función de adherirse a un determinado receptor, mientras otras partes de la misma jueguen distintos papeles.

  El desafío de la detección de comunidades es inferir estructuras de comunidades latentes dada solo el gráfico de entrada $G = (V, E)$.

- **Clasificación, regresión y agrupamiento de grafos**: Por ejemplo, dado un grafo molecular, se podría querer construir un modelo de regresión que pueda predecir la toxicidad o solubilidad de esa molécula. En estas aplicaciones de clasificación o regresión de grafo, se busca aprender sobre grafos, pero en lugar de hacer predicciones sobre los componentes individuales de un solo grafo (es decir, los nodos o las aristas), se da un conjunto de datos de múltiples grafos diferentes y nuestro objetivo es hacer predicciones independientes específicas para cada grafo.

  De todas las tareas de aprendizaje automático en grafos, la regresión y clasificación de grafo son quizás los análogos más directos del aprendizaje supervisado estándar. Cada grafo está i.i.d., lleva asociado una etiqueta, y el objetivo es utilizar un conjunto etiquetado de puntos de entrenamiento para aprender asignar etiquetas a grafos. De manera similar, el agrupamiento de grafos es la extensión directa del agrupamiento no supervisado para datos de grafos. Sin embargo, el desafío en estas tareas consiste en definir características útiles que tengan en cuenta la estructura relacional del grafo.

Este trabajo está enfocado en este último tipo de problema, en la clasificación de grafos. Además, se partirá de un conjunto de entrenamiento con etiquetas, por lo que la clasificación será supervisada.

5.2. Modelos de aprendizaje automático sobre grafos

Una vez se han detallado los tipos de tareas sobre grafos, en esta sección se van a exponer los modelos de aprendizaje automático que toman a los grafos como dato de entrada. De hecho, una de las ramas más activas de investigación en la problemática de predicción o clasificación de grafos, nodos o vértices es la de GNN (Redes Neuronales en Grafos). Dentro de esta se encuentra la GCN (Redes convolucionales en Grafos) y ambos conceptos están relacionados con los algoritmos MP (Propagación de Mensajes). Dada su impacto en tareas de clasificación de moléculas se va a definir cada uno de los conceptos mencionados y tratar alguno de los trabajos con mejores resultados.

Las **GNN** son un término amplio que abarca diversas arquitecturas de redes neuronales. A diferencia de las redes neuronales tradicionales, que están diseñadas para aprender de datos en formato de matriz o tensor, las GNN están diseñadas para aprender de datos en forma de grafo, donde los nodos representan entidades y las aristas representan las relaciones entre ellas. Se basan en la propagación de información a través del grafo utilizando diferentes esquemas de propagación de mensajes. Estos esquemas actualizan las características de los nodos en función de las características de sus vecinos. En cada iteración, se agregan las características de los nodos vecinos para actualizar las características del nodo actual, y este proceso se repite varias veces, permitiendo que la información se propague a través del grafo. A diferencia de los GCN, que utilizan una operación similar a la convolución, los GNN pueden manejar grafos más complejos que no necesariamente exhiben una estructura regular requerida para los GCN. Sin embargo, en el caso de los grafos de moléculas, generalmente no representan tales dificultades y se puede trabajar con GCN.

Una **CNN** (*Convolutional Neural Network*) es una red neuronal que utiliza operaciones de convolución para extraer características, típicamente, de imágenes. Una convolución es una operación matemática de dos funciones f y g que produce una tercera $(f * g)$ que expresa cómo la forma de uno es modificado por el otro. Por poner un ejemplo que se mencionará más tarde, las transformadas de Fourier discretas en el tiempo (*DTFT discrete-time Fourier Transform*) se pueden convolucionar de la siguiente forma para las secuencias x e y:

$$x * y = \mathrm{DTFT}^{-1}[\mathrm{DTFT}\{x\} \cdot \mathrm{DTFT}\{y\}]$$

En las imágenes, solo se desplazan *kernels* que se puedan aprender sobre la estructura de malla que forman los píxeles, extrayendo así la información más relevante. También se puede visualizar como la combinación de información en un píxel de sus píxeles vecinos. En los grafos se extiende esta idea a la concentración en un nodo a de la información de los nodos adyacentes. La forma en la que la información se transmite de nodo a nodo se llama Propagación de Mensajes.

Un **GCN** es una fusión entre una CNN y una GNN, dicho de otra forma, una CNN que trabaja sobre grafos. Esta idea de combinación de redes, surge del

éxito que han tenido las redes neuronales profundas con capas de convolución en conjuntos de datos que se representan en el espacio euclidiano (imágenes, p. ej.) y de las limitaciones que tienen estas para comprender datos en forma de grafos. La GCN utiliza una operación similar a la convolución para extraer características de un grafo. Generalmente, la operación se define en términos de la matriz de adyacencia del grafo y las características de entrada de cada nodo en el grafo. Las características de salida de cada nodo se obtienen agregando las características de sus nodos vecinos, ponderadas por la matriz de adyacencia. El proceso se repite para varias capas, lo que permite que la red aprenda representaciones estructurales del grafo. En la imagen 5.1 se puede observar el modelo GCN aplicado a la clasificación de regiones cerebrales. Los nodos del grafo generado cuentan con un vector de características con una dimensión inicial de 128 parámetros. Este grafo se va simplificando a través de las tres capas de convolución que se pueden observar, hasta conseguir una representación adecuada para que la red neuronal pueda aprender.

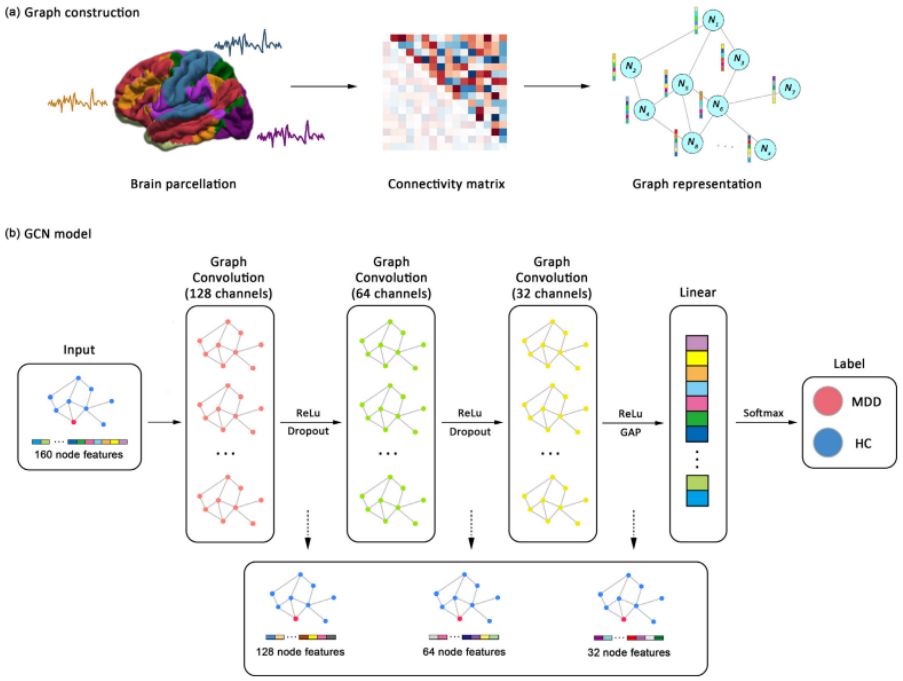


Figura 5.1: Modelo de GCN para clasificación de regiones cerebrales extraído de [Qin+22]

Se han intentado muchos enfoques para trabajar con GCN. pero se pueden subdividir en dos grupos: los espectrales y los espaciales. En el enfoque espectral se definen las operaciones de convolución en transformadas de Fourier sobre grafos. Para los métodos espaciales, las operaciones de convolución de grafos se diseñan mediante la agregación de las representaciones de nodos directamente desde su vecindario. La mayoría de los métodos mencionados involucran la transformación, propagación y agregación de características de nodos a través del gráfico, que se

recogen en la arquitectura de paso de mensajes.

5.2.1. Agrupación jerárquica de grafos con aprendizaje estructural

Una de las técnicas en el *dataset* PROTEINS de mayor éxito, con una tasa de precisión del 84,91 %, ha sido la de la agrupación jerárquica de grafos con aprendizaje estructural [Zha+19b]. Este trabajo está enfocado en las operaciones de "puesta en común" o *pooling*, en vez de las más tradicionales operaciones convolucionales. A su vez, esta estrategia se basa en otra más general y que es bastante recurrente en los distintos trabajos de clasificación de grafos. Esta arquitectura más general a la que me refiero es la arquitectura de paso de mensajes (*Message Passing*). Este trabajo recoge bien el diseño y varios resultados interesantes que se han obtenido [Gil+17], además, fue pionero en identificar y describir la arquitectura. Esta se utiliza en redes neuronales con el nombre MPNN del inglés *Message Passing Neural Networks*.

La parte importante, donde se va a actualizar la información de la red neuronal, se llama "paso hacia adelante" y tiene dos fases: una fase de paso de mensajes (o preactivación) y una fase de lectura (o activación).

- La fase de paso de mensajes se ejecuta durante T pasos de tiempo y se define en términos de funciones de mensaje M_t y funciones de actualización de vértice U_t. Durante la fase de paso de mensajes, los estados ocultos h_v^t en cada nodo del gráfico se actualizan en función de los mensajes m_v^{t+1} según:

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

$$h_{t+1}^v = U_t(h_t^v, m_{t+1}^v)$$

  donde en el sumatorio $N(v)$ denota el vecindario de v en el grafo G.

- En la fase de lectura se computa un vector de características para todo el grafo utilizando una función de lectura R de acuerdo a

$$\hat{y} = R(\{h_T^v \mid v \in G\})$$

Las funciones de mensaje M_t, funciones de actualización de vértices U_t y la función de lectura R son todas funciones diferenciables aprendidas. R opera en el conjunto de estados de los nodos y debe ser invariante a las permutaciones de los estados de los nodos para que el MPNN sea invariante al isomorfismo de grafos.

5.3. Persistencia homológica en clasificación molecular

A diferencia de los modelos de la anterior sección que trabajan con grafos, esta técnica [ZW19], que tiene los mejores resultados en la clasificación de moléculas de la base de datos NCL1, hace un preprocesado especial de los datos. Como se verá, este método partirá también del descriptor grafo molecular, pero se van a hacer distintas variaciones hasta conseguir un descriptor molecular muy eficaz llamado imagen de persistencia y, finalmente, esta será el que se utilice de dato de entrada en este modelo de aprendizaje automático.

5.3.1. Modelo general de la persistencia homológica para clasificación de grafos

En líneas generales, teniendo una función, $f : X \to \mathbb{R}$ la persistencia homológica resume las propiedades de distintas escalas o resoluciones de X simultáneamente en un único resumen llamado diagrama de persistencia. La técnica de persistencia homológica es especialmente interesante porque se puede aplicar a tipos de datos complejos como pueden ser objetos 3D o grafos. Por esta razón, este método se ha convertido muy popular en la ciencia de datos para vectorizar datos complejos.

Ahora procedo a dar una explicación más detallada, sea X una forma (en nuestro caso un grafo) y sea una secuencia de subconjuntos ascendentes de X : $X_1 \subseteq X_2 \subseteq ... \subseteq X_n = X$ llamada **filtración** de X. En el párrafo anterior, ha sido referido como **escala** o **resolución** a cada uno de estos subconjuntos que conforman la filtración. Lo interesante de esta técnica viene al observar la aparición de ciertas propiedades en una resolución X_i y la desaparición de las mismas en otra escala X_j. Estas propiedades vienen descritas por las clases homológicas. Las clases homológicas proveen un lenguaje matemático para los agujeros de un espacio topológico. La creación y desaparición de estas propiedades se puede capturar en un diagrama de persistencia.

En concreto, dado un grafo $G = (V, E)$ con V vértices y E aristas sea un **descriptor** f:

$$f : V \to \mathbb{R}$$
$$f(u) \mapsto \text{grad}(u)$$

Con $grad(u)$ se denota al grado del vértice, es decir, al número de vecinos que este tiene. f se puede extender a E describiendo una arista por los vértices que la delimitan,

$$f : E \to \mathbb{R}$$
$$f(u, v) \mapsto \max\{f(u), f(v)\}$$

es decir, el grado más alto de los nodos del vértice. Ahora se puede definir una filtración del grafo de esta forma,

$$G_{\leq a} = \{\sigma \in V \cup E | f(\sigma) \leq a, a \in \mathbb{R}\}$$

Que es el conjunto compuesto por las aristas cuyo nodo de grado más alto es menor o igual a "a" y por los nodos cuyo grado es menor o igual a "a". De la misma manera, se puede extender el descriptor f a V mediante:

$$\begin{aligned} f : V &\to \mathbb{R} \\ f(u) &\mapsto \min_{u \in e, e \in E}(f(e)) \end{aligned}$$

Que es el mínimo de los máximos grados de las aristas que tengan a "u" como nodo. Cuando se escanea G mediante las filtraciones definidas anteriormente, se crearán y unirán componentes conexos y nuevos ciclos serán formados. Las características 1-dimensionales (ciclos) no serán destruidos y persistirán en todo el escaneo, dando lugar a la conocida como persistencia extendida.

Un **diagrama de persistencia** consiste en un conjunto de puntos en el plano, donde uno de los ejes corresponde al tiempo de nacimiento y el otro al tiempo de muerte de una propiedad topológica de la estructura. Así, el punto $p = (b, d)$ b de *birth* y d de *death*, indicará que una propiedad determinada "nace" en el tiempo b y "muere" en el tiempo d. Hay un conjunto de distancias que se utilizan para estos diagramas, entre ellos se incluyen la distancia "cuello de botella" y la de Wasserstein. En las siguientes líneas voy a explicar en qué consisten las dos.

La distancia **cuello de botella** mide la similitud entre dos diagramas de persistencia. Sean dos diagramas de persistencia A y B con sus correspondientes puntos:

$$A = \{a_1, a_2, \ldots, a_n\} \text{ y } B = \{b_1, b_2, \ldots, b_m\} \text{ donde } a_i = (p_i, q_i) \in \mathbb{R}^2 \text{ y } b_j = (p_j, q_j) \in \mathbb{R}^2$$

Para calcular esta distancia, se deben seguir varios pasos. En primer lugar, se tienen que emparejar los puntos, sin embargo, se puede observar que los dos diagramas no tienen por qué tener el mismo número de puntos. Por ello, se introduce el concepto de emparejamiento parcial entre los dos conjuntos A y B que consiste en una función biyectiva entre subconjuntos de A y B. El segundo paso, consiste en el costo de emparejamiento. Dado un emparejamiento parcial M, el coste de M denotado como $C(M)$, se define de la siguiente manera:

$$\begin{aligned} C(M) = \text{máx}\{ & \\ & \sup\{||a_i - M(a_i)||_\infty\}, \\ & \sup\left\{\frac{1}{2}(q_i - p_i) \mid (p_i, q_i) \text{ desparejado en } A\right\}, \\ & \sup\left\{\frac{1}{2}(q_j - p_j) \mid (p_j, q_j) \text{ desparejado en } B\right\} \\ & \} \text{ con } a_i \in k \subset A. \end{aligned}$$

Cabe aclarar que para los puntos desparejados se está calculando el punto más cercano de la diagonal. En el tercer paso, se calcula la distancia cuello de botella en sí, que se denota como $D(A, B)$ y se define como $D(A, B) = inf_{M:A \to B} C(M)$.

La distancia de **Wasserstein**, conocida también como la **distancia del movedor de tierra**, se utiliza para medir la distancia entre dos distribuciones de probabilidad sobre una región. Informalmente, si se consideran las distribuciones como montones de tierra, la distancia de Wasserstein sería el costo mínimo para convertir un montón en el otro. Para el caso que concierne, esta distancia viene definida como:

$$W_q(X,Y) = \inf_{\eta: X \to Y} \sum_{x \in X} ||x - \eta(x)||_\infty$$

donde X e Y son los diagramas de persistencia y η es un emparejamiento perfecto entre los intervalos. El signo de infinito se debe a que se han incluido los puntos de la diagonal para asegurarnos de que el emparejamiento perfecto exista.

En concreto, teniendo un conjunto de grafos moleculares, se podría convertir cada grafo a una representación basada en la persistencia. Se partiría de un diagrama de persistencia, realizando unos cambios se tendrían los correspondientes puntos en un espacio de estados de persistencia y, después, se medirían unas distancias o se aplicaría un *kernel* apropiado que sirviese en tareas de clustering.

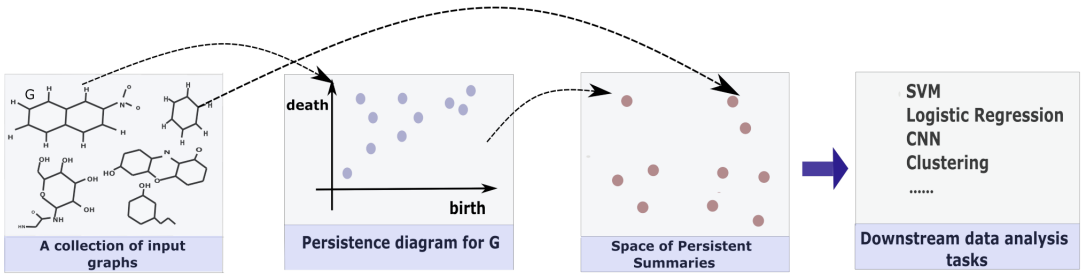


Figura 5.2: *Framework* de análisis de datos basados en persistencia, fuente: [ZW19]

Sin embargo, este espacio de estados persistentes no es el más adecuado para técnicas de aprendizaje automático. Por ello, hay que aplicar otro proceso de vectorización al diagrama de persistencia, entre estos procesos se encuentran *Persistence Scale-Space kernel*, imágenes de persistencia, *Persistence Weighted Gaussian kernel* (PWGK), *Sliced Wasserstein kernel* y *Persistence Fisher kernel*.

En estos enfoques, cuando se calcula el *kernel* o las distancias entre estados persistentes, la importancia de distintas propiedades persistentes es predeterminada. En las imágenes de persistencia y el PWGK, la importancia de tener una función de peso en el plano del diagrama de persistencia está incluida en la formulación de los kerneles. De todas formas, la función de peso necesita estar preestablecida. También es importante mencionar que la elección de la función de peso depende de la naturaleza de los datos.

Para generar esa vectorización necesaria para tareas de aprendizaje automático, se parte de un diagrama de persistencia A. Haciendo una transformación

lineal $T : \mathbb{R}^2 \to \mathbb{R}^2$ donde $T(x, y) = (x, y - x)$ se tiene el **plano de persistencia**. Ahora, sea $\Phi_u : \mathbb{R}^2 \to \mathbb{R}$ una distribución de probabilidad diferenciable con media $u \in \mathbb{R}^2$ (por ejemplo, la distribución Gaussiana normalizada). Teniendo esto en cuenta, sea una distribución de peso $\alpha : \mathbb{R}^2 \to \mathbb{R}$ no negativa para el plano de persistencia $T(A)$. Su **superficie de persistencia** $\rho_A : \mathbb{R}^2 \to \mathbb{R}$ viene definido como:

$$\rho_A(z) = \sum_{u \in T(A)} \alpha(u)\phi_u(z) \qquad \forall z \in \mathbb{R}^2$$

La **imagen de persistencia** es una discretización de la superficie de persistencia. Concretamente, se subdivide una región cuadrada en un conjunto P de N rectángulos más pequeños, de la misma forma que una imagen se subdivide en píxeles. La imagen de persistencia $PI(A)$ del diagrama A se define como $PI_A = \{PI[p]\}_{p \in P}$, conjunto de N números (uno por cada píxel), donde cada número viene determinado por $PI[p] = \int \int_p \rho_A dydx$. La imagen de persistencia se puede ver como un vector en $\mathbb{R}^N$ y podría servir para calcular la distancia entre diagramas de persistencia mediante $||PI_1 - PI_2||_2$.

Si la distancia entre dos imágenes de persistencia es pequeña, se podría decir que los grafos representados se parecen y, por tanto, las moléculas tengan las mismas propiedades químicas.

5.3.2. WKPI

Weighted Persistence Image Kernel, es una manera concreta de aplicar la homología topológica que ha dado muy buenos resultados en la clasificación de grafos en los *datasets* anteriormente citados. En las siguientes líneas se explicará cómo se aplica. Se define un *kernel* para las imágenes de persistencia: Sea $\omega : \mathbb{R}^2 \to \mathbb{R}$ una función de peso y p_s el centro del s-ésimo pixel de la imagen de persistencia (que es un vector en $\mathbb{R}^N$). Dados dos imágenes de persistencia PI y PI' el **kernel de peso** (*weighted kernel*) de la imagen de persistencia se define como:

$$k_\omega(PI, PI') := \sum_{s=1}^{N} \omega(p_s) e^{-\frac{(PI(s) - PI'(s))^2}{(2\sigma^2)}}$$

La función de peso se puede escoger entre distintas clases de funciones como la mezcla de m Gaussianas de 2 dimensiones o los polinomios de d-grado de dos variables.

La **WKPI distancia** entre dos diagramas A y B cuyas imágenes de persistencia son PI_A y PI_B se define como:

$$D_\omega(A, B) \coloneqq \sqrt{k_\omega(PI_A, PI_A) + k_\omega(PI_B, PI_B) - 2k_\omega(PI_A, PI_B)}$$

Volviendo a la visión general del problema, se tiene una colección Ξ de objetos (grafos) $\{X_1, \ldots, X_n\}$ (que representan moléculas). Estos grafos van a estar etiquetados en k clases distintas $\{C_1, \ldots, C_k\}$, en el *dataset* PROTEINS citado anteriormente, por ejemplo, $k = 2$ ya que están etiquetadas en si son enzimas o

no. Se calculan los diagramas de persistencia $\{A_1, \ldots, A_n\}$ en base a una filtración que se detallará más adelante.

Cuando se hace la WKPI distancia entre dos diagramas en base a una función de peso ω, en realidad se quiere ver cuánto se asemejan los dos grafos, es decir, se busca $D_\omega(X_i, X_j) \approx D_\omega(A_i, A_j)$. Para ello, nuestro objetivo es aprender una buena métrica de distancia para el *dataset*. Esto se formulará como aprender una mejor función de peso ω^* tal que su WKPI distancia inducida acierta en la clase de etiquetado de los grafos. Expresado matemáticamente:

$$cost_w(t,t) = \sum_{i,j \in C_t} D_\omega^2(A_i, A_j)$$

y

$$cost_w(t,\cdot) = \sum_{i \in C_t, j \in 1,2,\ldots,n} D_\omega^2(A_i, A_j)$$

La primera de ellas, $cost_w(t,t)$, calcula la diferencia que hay entre los grafos de la misma clase. Mientras que en $cost_w(t,\cdot)$, se calcula la diferencia entre los grafos de una clase y el resto de grafos del *dataset*. Una buena métrica debería dar una pequeña distancia en la primera y mucha distancia en la segunda.

Se define nuestro problema de optimización, sea $\omega : \mathbb{R}^2 \to \mathbb{R}$ una función de peso, el **coste total de la distancias WKPI inducidas** sobre el conjunto Ξ está definido como:

$$TC(\omega) \coloneqq \sum_{t=1}^{k} \frac{cost_w(t,t)}{cost_w(t,\cdot)}$$

El problema de distancia óptima tiene como objetivo encontrar la mejor función de peso ω^* de una clase de funciones F tal que el coste total sea mínimo. Es decir, $TC^* = \min_{\omega \in F} TC(\omega)$ y $\omega^* = \arg\min_{\omega \in F} TC(\omega)$.

Para dar solución a este problema de optimización, conviene representarlo en modo matricial porque muchos lenguajes de programación, como Python, realizan las operaciones de forma más eficiente así.

Sea la matriz $L = G - \Lambda$, donde $\Lambda = [\Lambda_{ij}]_{n \times n}$ con $\Lambda ij = D_\omega(A_i, A_j)$ para $i, j \in \{1, 2, \ldots, n\}$ es la matriz de distancia de los objetos $X_1, X_2, \ldots, X_n$ y donde la matriz G se define como:

$$G = [g_{i,j}]_{n \times n} \text{ donde } g_{i,j} = \begin{cases} \sum_{l=1}^{n} \Lambda_{il} & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

Por razones que no se van a detallar el coste total se puede representar también como $TC(\omega) = k - \mathrm{Tr}(HLH^T)$ y el problema de distancia óptima como $\min\limits_{n} \omega(k - \mathrm{Tr}(H_\omega L_\omega H_\omega^T))$ sujeto a $H_\omega L_\omega H_\omega^T = I$.

Para resolver este problema, el de encontrar una función de peso óptima ω^*, se ha recurrido al descenso estocástico de gradiente. En primer lugar, se calculan los diagramas de persistencia de los distintos grafos etiquetados utilizando un criterio apropiado de filtración. Después se encuentran los mejores parámetros

de ω^*, que van a depender de la clase de funciones que se hayan escogido. Por ejemplo, si ω es de la clase de funciones de m mezclas de funciones de peso no negativas Gaussianas se tiene $4-m$ parámetros $\{x_r, y_r, \sigma_r, w_r \mid r \in \{1, 2, \ldots, n\}\}$ con

$$\omega(z) = \sum_{r=1}^{m} w_r e^{-\frac{(z_x - x_r)^2 + (z_y - y_r)^2}{(\sigma_r^2)}}.$$

En este caso, se pueden seguir varias estrategias para inicializar el centro de las gaussianas: una aleatoria, otra aplicando el algoritmo k-means para identificar los centros u otra aplicando el algoritmo de k-centros para identificar m centros.

En el trabajo de referencia se utiliza el esquema de actualización de parámetros de Armijo-Goldstein para cada paso del descenso de gradiente, pero se podría utilizar cualquier otro esquema utilizado en estos algoritmos de *backtracking*.

El procedimiento de optimización se detiene cuando el coste de las funciones converge o el número de iteraciones supera un límite. La complejidad algorítima es de $O(Rs^2N)$ donde R es el número de iteraciones, s es el tamaño del paso de descenso de gradiente y N es el número de píxeles de una imagen de persistencia.

Una vez aprendida ω^*, se pasa a la clasificación de grafos utilizando el *kernel* SVM (*Support Vector Machines*). El SVM es un modelo de aprendizaje supervisado con un algoritmo asociado, en este caso, con el ω^* WKPI *kernel* aprendido.

Capítulo 6

Datasets

6.1. PROTEINS

La base de datos PROTEINS fue creada en 2003 por Dobson y Doig y fue introducida en este trabajo [Bor+05]. Contiene 1113 proteínas etiquetadas en si son encimas (59 %) o no lo son (41 %). La función de predicción en este conjunto de proteínas es particularmente difícil porque Dobson y Doig eligieron proteínas cuyas cadenas no se alineaban con ninguna otra cadena en el *dataset* con una puntuación Z de 3.5 o más fuera de las estructuras de sus padres.

Adicionalmente, se utilizarán unas transformaciones de este *dataset* que se recogen en [ZW19]. Estas transformaciones están documentadas en el método WKPI descrito 5.3.1 ahí. Concretamente, se recogen las imágenes y diagramas de persistencia. Las imágenes de persistencia tienen una resolución de 20×20 píxeles y son unicanales. Respecto a los diagramas se puede decir que no tienen el mismo número de puntos. Cabe destacar que las imágenes y diagramas de persistencia no están vinculadas biyectivamente con el *dataset* de grafos PROTEINS original, es más, el *dataset* de los grafos contiene una molécula más que sus transformaciones.

Las moléculas en todas sus representaciones vienen etiquetas en si se corresponden a una encima o no, concretamente, utilizando un número entero para esto.

6.2. NCI1

El NCI1 es una base de datos de compuestos químicos (representados como grafos) clasificados en si son activos o no en contra de células cancerígenas "no pequeñas" de pulmón. Fue introducida en este trabajo [WK06] por Nikil Wale y George KarypisEn en el año 2006. En concreto, contiene 4110 grafos con 29,87 vértices y 32,3 aristas de media. Este conjunto de datos está equilibrado, tiene un 50 % de elementos en cada una de las dos clases.

6.3. Tox21 AhR

El Tox21 (*Toxicology in the 21st Century)* es un conjunto de bases de datos públicas creadas para el reto *Tox21 Data Challenge 2014* [Adv14]. Este desafío fue diseñado para ayudar a los científicos a entender el potencial de las sustancias químicas y compuestos que se están probando a través de la iniciativa "Toxicología en el siglo XXI". El objetivo del desafío es incentivar el análisis de datos en investigadores independientes para revelar cuán bien pueden predecir la interferencia de los compuestos en las vías bioquímicas utilizando únicamente datos de estructura química. Los modelos computacionales producidos fueron convertidos en herramientas de toma de decisiones para las agencias gubernamentales para determinar qué productos químicos, ambientales y medicamentos son potencialmente más preocupantes para la salud humana.

La base de datos concreta que se va a utilizar en este trabajo es la Tox21 AhR e incluye, 8169 moléculas etiquetadas en si interfieren o no en las vías bioquímicas hepáticas. Este conjunto de datos está desequilibrada ya que contiene un 88,3 % de los elementos en un grupo. Los descriptores que se han encontrado son los grafos moleculares y los SMILES.

6.4. MUTAG

El conjunto de datos MUTAG, descrito por primera vez en este trabajo [**mutag\textit {Dataset}**], contiene 188 componentes nitroaromáticos etiquetados según sus "efectos mutagénicos en una bacteria gram negativa específica (*Salmonella typhimurium*)". El conjunto de datos original incluye 188 grafos con un promedio de 18 nodos y 20 aristas para cada grafo. Los nodos del grafo tienen 7 etiquetas y cada grafo está etiquetado como perteneciente a una de las dos clases. Estas clases no están equilibradas, ya que dos tercios pertenecen a un grupo.

Por homogeneizar las bases de datos, se ignorará la información correspondiente al etiquetado de nodos del grafo. Se han podido encontrar varios descriptores moleculares para este conjunto de datos: los grafos, los diagramas de persistencia, las imágenes de persistencia y los SMILES.

En la siguiente tabla 6.1 se pueden observar varios datos de los *datasets* escogidos. Núm. es la abreviatura de Número, Prom. la de Promedio y D. la de Descriptor molecular; con "Clases" se refiere al número de grupos que existen en el *dataset* (enzimas o no enzimas, por ejemplo):

	Núm. Grafos	Clases	Prom. Nodos	Prom. Enlaces	D. Grafo	D. Diagrama	D. Imagen	D. SMILES
MUTAG	188	2	17.93	19.79	✓	✓	✓	✓
NCI1	4110	2	29.87	32.30	✓	X	X	X
Tox21 AhR	8169	2	18.09	18.50	✓	X	X	✓
PROTEINS	1113	2	39.06	72.82	✓	✓	✓	X

Tabla 6.1: Características de los *Dataset*s

	D. Grafo	D. Imagen	D. SMILES
MUTAG	HGP-SL / GCNConv	WKPI / SVM	RNN
NCI1	HGP-SL / GCNConv	X	X
Tox21 AhR	HGP-SL / GCNConv	X	RNN
PROTEINS	HGP-SL / GCNConv	WKPI / SVM	X

Tabla 6.2: Técnicas de aprendizaje automático empleadas

Capítulo 7

Estudio comparativo entre distintos descriptores y técnicas de aprendizaje automático

El objetivo de esta sección es comparar el desempeño de distintos descriptores y técnicas de aprendizaje automático a la hora de clasificar moléculas. De esta forma se evaluará cuál es la mejor estrategia para clasificar grafos moleculares y se discutirán también las limitaciones. Los modelos aplicados se muestran en la anterior tabla 6.2.

Para hacer una comparativa válida, se han escogido *datasets* de forma que todos los descriptores tengan al menos dos conjuntos de datos distintos. Por otra parte, se discutirán al menos dos técnicas de aprendizaje automático por cada descriptor molecular con el objetivo de no solo evaluar el modelo de aprendizaje, sino también la utilidad del descriptor. Concretamente, uno de los modelos de aprendizaje automático será el que mejores resultados haya obtenido con el descriptor molecular correspondiente en uno de los datasets, mientras que la otra técnica será otro modelo escogido para la ocasión por razones que se indicarán en cada caso. De esta forma, se podrá evaluar los descriptores con mayor robustez y determinar las mejoras de distintos modelos de *Machine Learning*.

Las técnicas dependerán de la naturaleza de los datos de entrada. Estos datos de entrada provienen de los *dataset* anteriormente expuestos. Adicionalmente, estos datos podrán sufrir transformaciones que se explicarán a la hora de comentar su desempeño.

Para el estudio comparativo, no se dispone de la información que vincula un grafo con su correspondiente transformación de persistencia (diagrama o imagen) en todos los *datasets*. Por ello, es necesario hacer una validación cruzada para poder comparar rendimientos. Una **validación cruzada** es una técnica para evaluar el rendimiento de un modelo predictivo, garantizando que es independiente de la partición de datos de entrenamiento y prueba.

Existen varios tipos de validación cruzada, quizás la más popular sea la validación cruzada de K iteraciones, donde se reserva un conjunto de entrenamientos

de testeo y con el resto se hacen varias iteraciones de entrenamiento. Sin embargo, como no podemos asegurar que el conjunto de prueba sea el mismo entre todos los descriptores, este método queda descartado. Tampoco se puede optar por el método más preciso, la conocida como "validación cruzada dejando uno fuera", donde solo se deja un dato de muestra para cada iteración, por ser computacionalmente demasiado costosa, ya que hay que hacer tantas iteraciones como el tamaño del *dataset* (número de moléculas). Por lo tanto, se ha optado por la validación cruzada aleatoria donde los conjuntos de datos de prueba y entrenamiento se escogen aleatoriamente. No tiene los inconvenientes que hemos mencionado para las otras dos tácticas, pero puede haber problemas de representación de la muestra porque los datos se pueden repetir o no salir.

Para ver el rendimiento de cada uno de los descriptores y técnicas, se han dividido los conjuntos de datos en subconjuntos de entrenamiento, validación y prueba.

- El **conjunto de entrenamiento** se nutre de un 80 % de las moléculas de los *dataset* como norma general. Durante el entrenamiento, el modelo aprende a asociar las características de las moléculas en forma de descriptores con sus correspondientes etiquetas. El modelo ajusta sus parámetros y encuentra la mejor forma de agrupar las moléculas en las distintas categorías.

- El **conjunto de validación** se utiliza para ajustar los hiperparámetros del modelo y evaluar su rendimiento durante el entrenamiento. Los hiperparámetros pueden ser el tipo de kernel, parámetros de regularización, etc. dependerán del modelo de aprendizaje automático escogido. Cabe destacar que este conjunto de validación puede ser un subconjunto del *dataset* (con un tamaño del 10 %), pero también se puede considerar que el conjunto de validación se derive del conjunto de datos completo mediante un proceso de división aleatoria que se realiza para generar los subconjuntos de entrenamiento y prueba. Este último caso es útil cuando hay que hacer un estudio comparativo por validación cruzada.

- El **conjunto de prueba** es utilizado para evaluar el rendimiento final del modelo después de completar el entrenamiento y la validación. Este conjunto de datos no se evalúa durante el entrenamiento, es un subconjunto independiente al de entrenamiento o validación y tiene un tamaño del 10 % o 20 % dependiendo de cómo es el conjunto de validación. El modelo, una vez entrenado, se utiliza para predecir las etiquetas de las imágenes en el conjunto de prueba. Luego se comparan estas etiquetas predichas con las etiquetas reales para calcular la precisión de la clasificación.

Para computar los cálculos se ha utilizado un ordenador convencional sin tarjeta gráfica dedicada y con un procesador Ryzen 7. Las limitaciones del hardware no han supuesto un problema para testear los distintos modelos, ya que los tiempos de ejecución han sido asumibles.

7.1. Aplicación de modelos de Aprendizaje Automático en Grafos Moleculares

7.1.1. Aplicación del modelo HGP-SL en Grafos Moleculares

Se comenzará con los descriptores en forma de grafos moleculares. El trabajo con mejores resultados en este *dataset* con este descriptor, la *Graph Convolutional Network* HGP-SL [Zha+19b], ha sido descrito anteriormente aquí 5.2 y será utilizado como trabajo de referencia con este descriptor. Como se ha mencionado, se ha subdividido el *dataset* en tres conjuntos: entrenamiento, validación y prueba. Se ha ejecutado el modelo para hacer una validación cruzada aleatoria con 10 iteraciones con cada uno de los cuatro *datasets* que se disponen. En estas ejecuciones el modelo se ha entrenado durante distintas épocas que han ido desde 120 a 260 (hasta que han convergido). Los resultados obtenidos han sido los siguientes, se indica la semilla para la división aleatoria de los conjuntos para facilitar la reproducibilidad. El tiempo se muestra en segundos. El código que ha servido de base para estos resultados se puede encontrar aquí [Zha+19a].

En esta primera tabla 7.1, se pueden encontrar los resultados para las 10 iteraciones del GNN HGP-SL sobre el *dataset* PROTEINS:

Semilla	**Tiempo Total**	**Pérdida**	**Precisión**
900	849.2281	0.5306	0.7679
876	635.0016	0.5154	0.732
800	1087.7732	0.5396	0.7600
666	372.6295	0.6531	0.6518
500	852.2336	0.5717	0.7589
456	338.5842	0.6164	0.7232
400	659.0777	0.4861	0.7679
57	491.8419	0.5246	0.7321
50	737.7268	0.5652	0.7857
777	713.4637	0.4219	0.8571
Media	**673.75 s**	**0.5425**	**0.7536**

Tabla 7.1: 10 ejecuciones sobre grafos moleculares en PROTEINS con la GNN HGP-SL

Cabe recalcar que la semilla 777 es la que da el mejor resultado y es utilizada en la parte de la conclusión de los investigadores que crearon el método. Sin embargo, una media del 75,35 % de tasa de acierto es un dato relativamente bueno en esta tarea, véase [AI18]. Con "Pérdida" se refiere a pérdida total en el conjunto de prueba. Es la suma acumulativa de las pérdidas individuales del conjunto de prueba. En líneas generales, se comparan las predicciones del modelo con las etiquetas reales correspondientes en el conjunto de prueba. Cuanto mayor sea la

discrepancia entre las predicciones y las etiquetas, mayor será la pérdida. Para calcularla se ha utilizado la pérdida de log-verosimilitud negativa, una función de coste típicamente usada en modelos de aprendizaje automático. La precisión, simplemente, calcula el porcentaje de acierto de las etiquetas predichas con las reales del conjunto de testeo.

En la siguiente tabla 7.2 vemos el resultado que han obtenido el algoritmo HGP-SL en los distintos *datasets*.

	Tiempo Total (s)	**Pérdida**	**Precisión**	**Desviación Típica**
PROTEINS	673.75	0.542472	0.7536	0.0494
MUTAG	23.06	0.6441	0.8000	0.0224
NCI1	942.71	0.5085	0.7580	0.0087
Tox21	1125.06	0.6822	0.8987	0.0031

Tabla 7.2: Medias de los resultados de HGP-SL en 10 iteraciones

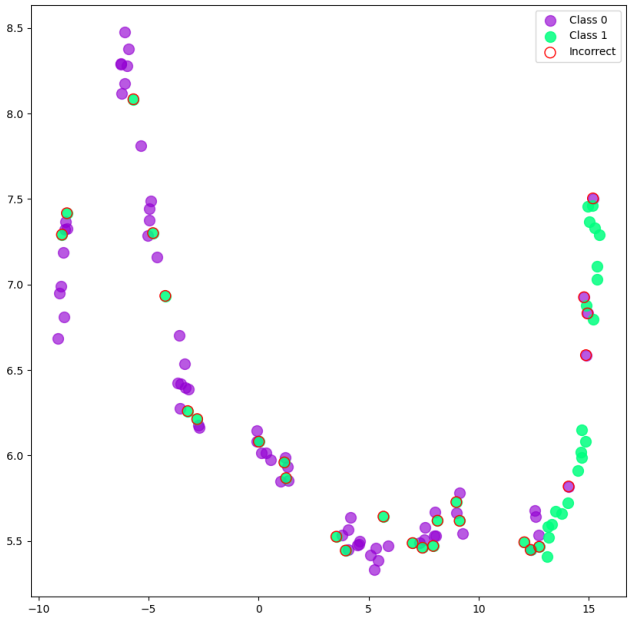
El tiempo total depende del número de grafos y el su tamaño, sin embargo, los valores obtenidos no son demasiado grandes. Cabe mencionar que en el *dataset* NCI1 se ha tenido que descartar un valor temporal por la detención de la ejecución.

Comparar los *datasets* por la pérdida puede ser engañoso por como se calcula la función de costo log-verosimilitud negativa, ya que esta puede depender del volumen de datos, entre otras inconveniencias. A pesar de ello, este dato será interesante para calcular los distintos modelos de aprendizaje sobre los mismos *datasets*.

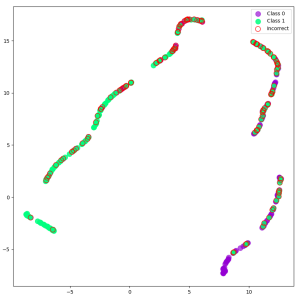
Finalmente, la precisión nos indica la tasa de acierto de las predicciones del modelo. En líneas generales, se ha obtenido una buena tasa, superior al 75 % en todos los casos. Cabe destacar el buen desempeño del modelo en el *dataset* Tox21 donde roza el 90 % de media en las 10 iteraciones que hemos realizado. La desviación estándar entre iteraciones es bastante baja, aunque es considerable en los *datasets* PROTEINS y MUTAG, esto se puede deber al pequeño número de moléculas de MUTAG o a la complejidad del *dataset* PROTEINS. En los otros dos datasets, en las muchas de las iteraciones se converge a resultados de la misma precisión, lo que explica la baja variabilidad.

En las siguientes imágenes 7.1, se observan los datos correctamente e incorrectamente clasificados para el conjunto de prueba en cada uno de los *datasets* que se dispone. El gráfico es una distribución de las características o *embeddings* que el modelo ha aprendido para cada molécula. El *embedding* es una representación vectorial del grafo que ha generado el modelo GCN, en concreto, son los valores en las neuronas de la última capa antes de la capa de salida. Para la obtención de las imágenes, se ha usado la técnica de reducción de dimensionalidad UMAP *Uniform Manifold Approximation and Projection*. Esta consiste en conservar las propiedades del conjunto en un espacio de menor dimensión. En concreto, para dibujar el diagrama se han coloreado las moléculas según su etiquetado real y

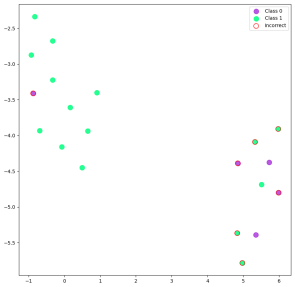
se han redondeado en rojo las que son un error de predicción. Esta técnica de visualización de datos se introdujo en 2018 en este trabajo [McI+18].



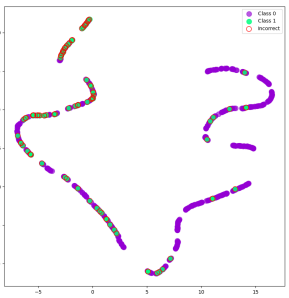
(a) PROTEINS



(b) NCI1



(c) MUTAG



(d) Tox21

Figura 7.1: Representación UMAP de la clasificación con HGP-SL

La interpretación de las representaciones UMAP se tiene que hacer con cautela porque pueden llevar a asunciones erróneas. Las agrupaciones de nodos que se pueden observar no necesariamente tienen que tener un significado. A pesar de ello, se puede advertir que en cada grupo se redondean de rojo más puntos de un color que del otro. Concretamente, se consideran erróneos más puntos verdes que violetas (excepto en MUTAG donde los errores están más repartidos). Esto indica que el modelo tiene un problema de desequilibrio de clases. Las razones

pueden ser varias, puede que haya pocos ejemplos de la clase mal clasificada o las características de ese grupo sean más difíciles de detectar. En la tabla 7.3 se puede observar numéricamente el desequilibrio de clases:

Dataset	**Precisión Clase 0 (%)**	**Precisión Clase 1 (%)**
PROTEINS	92.75	48.84
NCI1	66.99	71.22
MUTAG	40.00	73.33
Tox21_AHR	97.32	31.48

Tabla 7.3: Tasa de precisión del modelo HGP-SL por clases

7.1.2. Aplicación del modelo GCNConv en Grafos Moleculares

Para comparar el modelo anterior se ha escogido otra red convolucional sobre grafos. En concreto, es el modelo GCNConv de la biblioteca *Pytorch Geometric* con dos capas ocultas de 128 y 64 unidades y otra de clasificación lineal final. La razón de escoger dos capas o niveles y no más es que el costo computacional crece exponencialmente, además, puede empeorar el desempeño [Abu+18]. Esta biblioteca implementa lo investigado en este trabajo [KW16].

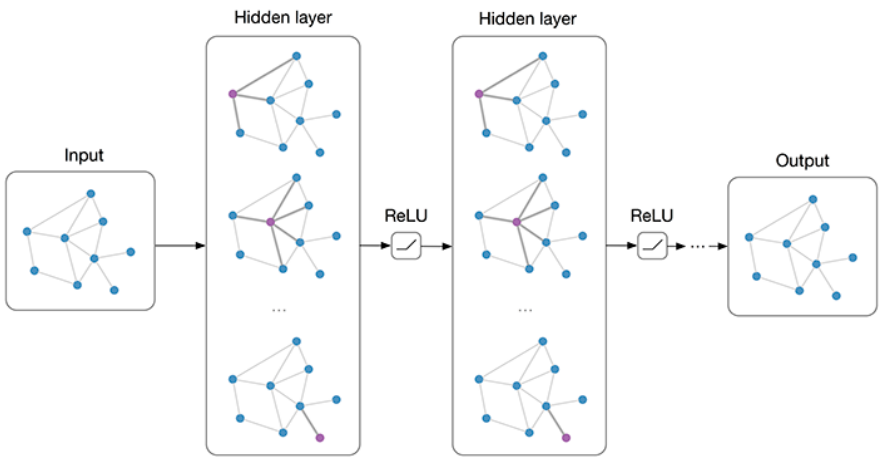


Figura 7.2: Esquema del modelo GCNConv

Este modelo, al igual que el anterior, pertenece a la familia de las Redes Convolucionales sobre Grafos. Una de las ventajas es que pueden incorporar el vector de características de los nodos. La mayor diferencia entre los dos métodos reside en que la el HGP-SL es un enfoque más complejo y reciente, ya que incluye componentes adicionales como *pooling* de grafos y aprendizaje de estructuras. En la siguiente imagen 7.2 se puede observar la estructura del modelo escogido para esta ocasión, creado por [Kip16].

Dataset	Pérdida	Precisión	Tiempo	σ (Precisión)
PROTEINS	0.619187	0.732143	292.506905	0.010310
NCI1	0.623917	0.659124	799.578424	0.007984
MUTAG	0.558568	0.700000	32.568241	0
Tox21_AHR	0.721651	0.881418	1438.935898	0

Tabla 7.4: Medias de los resultados de GCNConv en 10 iteraciones

En la siguiente tabla 7.4, se puede ver el desempeño de GCNConv. Se puede observar que las precisiones varían mucho entre distintos *dataset*, ya que van desde el 65, 9 %, un rendimiento bastante bajo, hasta el 88, 1 % que es un resultado muy bueno. Por tanto, este modelo tiene una tasa de generalización baja. Sin embargo, la desviación típica es pequeña, por lo que la sensibilidad a los datos de entrada no es muy grande.

En las imágenes 7.3, se observan las representaciones aprendidas de los modelos en dos dimensiones utilizando la técnica UMAP. Como en el caso anterior, en verde y lila se dibujan las clases reales y en rojo los errores de predicción. Los puntos también son representaciones en 2D de los *embeddings* de los datos en la última capa de la GCN.

Al igual que con el modelo HGP-SL, este modelo también hace una clasificación desbalanceada. En la siguiente tabla 7.5 se va a mostrar numéricamente el resultado:

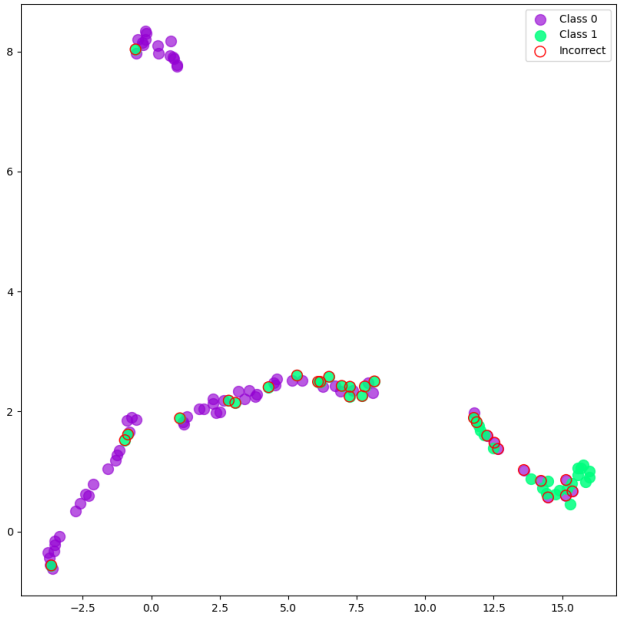
Dataset	**Label 0**	**Label 1**
PROTEINS	87.50 %	50.00 %
NCI1	74.02 %	55.07 %
MUTAG	55.56 %	90.91 %
Tox21 AhR	100.00 %	0.00 %

Tabla 7.5: Precisión por clase en distintos *datasets* con GCNConv

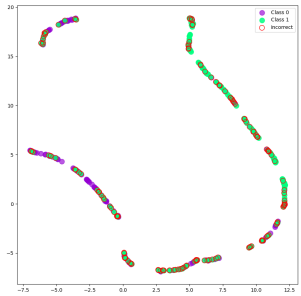
El desempeño varía mucho entre los conjuntos de datos. Por un lado, en el *dataset* Tox21, se ve una clasificación totalmente fallida, ya que ha metido todas las moléculas en un solo grupo. Por otro lado, en NCI1 la clusterización ha sido más balanceada, pero menos precisa. En PROTEINS y MUTAG el modelo peca de hacer un grupo bien a costa de sacrificar el desempeño en el otro. Por tanto, este modelo no tiene solo problemas de balanceo, sino que también presenta inconvenientes de generalización entre distintos conjuntos de datos.

7.1.3. Comparativa entre HGP-SL y GCNConv

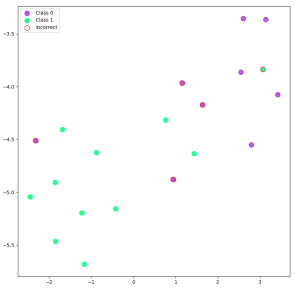
Para terminar con la comparativa del descriptor grafo molecular, se van a contraponer los resultados obtenidos con los dos modelos utilizados en la siguiente tabla 7.6. En ella se pueden observar las medias de los parámetros que se han



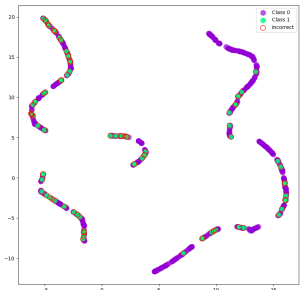
(a) PROTEINS



(b) NCI1



(c) MUTAG



(d) Tox21

Figura 7.3: Representación UMAP de la clasificación con GCNConv

medido en las 10 iteraciones. Ambos modelos se han entrenado en cada iteración con los mismos conjuntos de entrenamiento, validación y prueba. De esta manera, se evita en gran medida la sensibilidad a los datos de entrada.

La pérdida tampoco es útil para comparar los modelos, ya que su valor depende del número de épocas que se han necesitado en el conjunto de entrenamiento (algo que es independiente del desempeño en el testeo). Sin embargo, puede resultar útil para comparar los desempeños entre distintos conjuntos de entrenamiento, validación y prueba.

En cuanto al tiempo total de ejecución, no queda claro que modelo tarda más

Dataset	Pérdida		Precisión		Tiempo Total (s)		σ (Precisión)	
	GCNConv	HGP-SL	GCNConv	HGP-SL	GCNConv	HGP-SL	GCNConv	HGP-SL
PROTEINS	0.619187	0.542472	0.732143	0.753572	292.506905	673.75	0.010310	0.04942
NCI1	0.623917	0.508456	0.659124	0.757908	799.578424	942.71	0.007984	0.00872
MUTAG	0.558568	0.644128	0.700000	0.800000	32.568241	23.06	0	0.02236
Tox21_AHR	0.721651	0.682159	0.881418	0.898655	1438.935898	1125.06	0	0.00270

Tabla 7.6: Media de resultados sobre 10 iteraciones con dos modelos GCN

con los datos de la tabla anterior. Por ello, se ha decidido realizar un diagrama de burbujas 7.4 donde se vea la posible correlación con el tamaño de los datos, número de grafos y número de nodos promedio por grafo (ejes x e y respectivamente); el tiempo de ejecución, representado con el tamaño de las burbujas y el modelo que se ha utilizado, señalizado con el color.

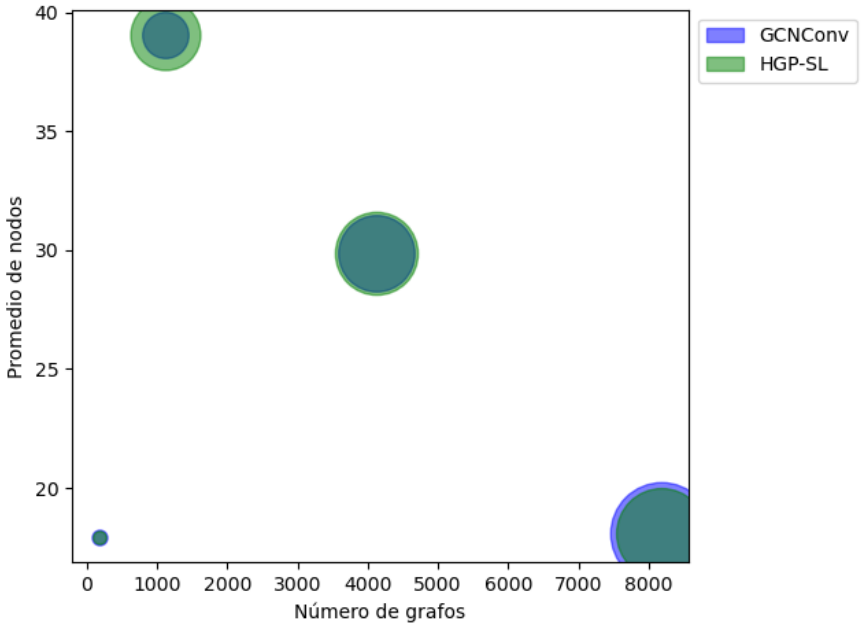


Figura 7.4: Tiempo de ejecución por tamaño de los modelos sobre grafos

Se pueden observar varias cosas interesantes. La primera es que no hay un modelo más rápido que otro, hay burbujas verdes más grandes que las azules y viceversa. La segunda, el tiempo de ejecución aumenta más con el tamaño de los *datasets* y no tanto con el promedio de nodos por grafo. La última se puede sugerir que el modelo HGP-SL procesa más rápidamente los *datasets* con el mismo número de grafos, pero con menor promedio de nodos. Por tanto, la velocidad de los modelos dependería del número promedio de nodos. Este fenómeno se puede deber a la técnica de *Graph Pooling* que incorpora el modelo HGP-SL. El *pooling* de grafos permite que el HGP-SL resuma la información de todo el grafo en una representación vectorial más compacta, en lugar de agrupar todos los nodos a la vez como lo hace el GCNCenv, lo que es más eficiente y depende directamente

del número de nodos.

En lo referente al balanceo de los modelos, ambos tienen un comportamiento similar, no se han observado diferencias significativas. Por tanto, los dos van a tener problemas notables de balanceo y su generalización no es buena.

En cuanto a la precisión de ambos modelos, el HGP-SL es mejor. El GCNConv no lo ha hecho mejor en ninguno de los datasets, pero en Tox21 y PROTEINS los resultados varían menos de un 2 %. En el resto, los resultados varían un 10 %, una diferencia considerable. En cuanto a la desviación típica de la precisión en las distintas iteraciones, se observa que HGP-SL tiene mayores valores. Por tanto, se puede considerar que el HGP-SL tiene mayor sensibilidad a los datos de entrada. Para finalizar, ambos modelos obtienen diferencias del 15 % en la precisión, comparando los *datasets* con mejor y peor desempeño. Por ello, se considera que la generalidad de estos modelos es limitada.

Antes de cerrar la sección, es importante recalcar que la generalización en estos conjuntos es muy complicada debido a la naturaleza de los datos moleculares. Por tanto, el peor desempeño en este aspecto no es achacable a las capacidades de los modelos. En todo caso, el parámetro más importante es la precisión y se han obtenido buenos resultados.

7.2. Aplicación de modelos de Aprendizaje Automático en Imágenes de Persistencia

El siguiente paso consiste en evaluar los *datasets* utilizando como descriptor molecular la imagen de persistencia. Esta imagen es una transformación del grafo que es propia de la técnica WKPI y se explica aquí 5.3.1. Básicamente, captura la información estructural del grafo (principalmente características homológicas) en una imagen de baja resolución (lo que facilita su procesamiento). Como hemos recogido en el capítulo de *Dataset*s, solo disponemos las bases de datos de MUTAG y PROTEINS para hacer la comparativa. Estas imágenes se han obtenido con el mismo método a partir de la representación en grafos de las moléculas y tienen las mismas características, por lo que la comparativa se puede llevar a cabo.

7.2.1. Aplicación del modelo WKPI en Imágenes de Persistencia

En esta sección se va a medir el desempeño de las imágenes de persistencia con el modelo WKPI. En este caso, el conjunto de validación servirá para ajustar los hiperparámetros del modelo, en concreto, el número k de la mezcla de las funciones Gaussianas y el valor σ_h (en esta sección se denota con el subíndice h de hiperparámetro para distinguirla de la desviación típica). El funcionamiento del modelo está explicada en esta sección 5.3.1.

Se han realizado dos experimentos, uno por *dataset*, con el código que se puede encontrar aquí. En los siguientes diagramas 7.5, 7.6 se pueden observar los resultados obtenidos según los parámetros k y σ_h.

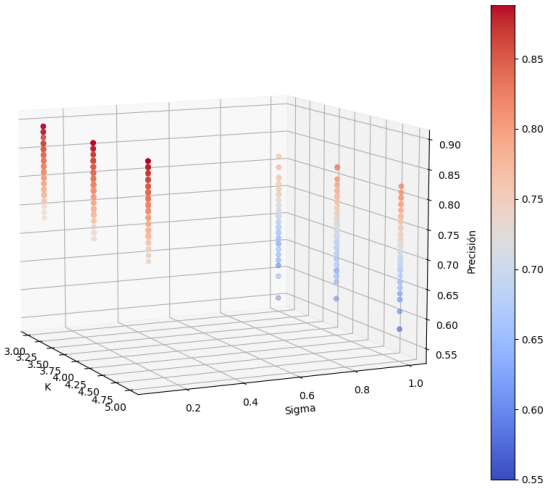


Figura 7.5: PROTEINS

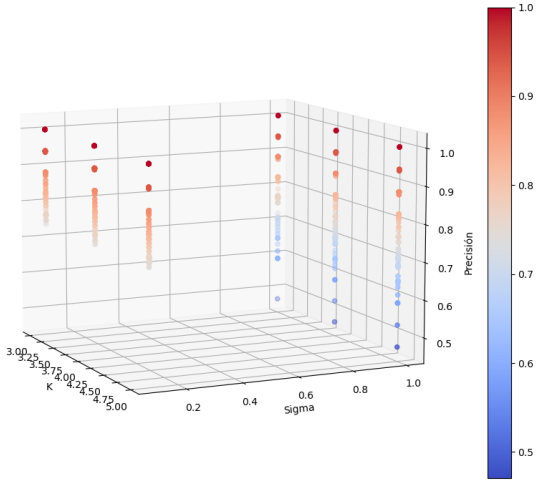


Figura 7.6: MUTAG

Figura 7.7: Precisión del modelo WKPI conforme a los parámetros k y σ_h

Como se puede observar, con $\sigma_h = 0{,}1$ se tienen resultados con menor variabilidad y mejores (en PROTEINS) o iguales (en MUTAG). El papel que juega el parámetro k parece más difícil de averiguar visualmente, parece que no es un factor determinante en el resultado de los datos.

Para que los resultados no dependan de las particiones que se hagan de los datos de entrada y facilitar la comparación con otros modelos, se han realizado 10 ejecuciones diferentes con distintos conjuntos de entrenamiento, validación y prueba. En esta ocasión se han elegido los mejores valores de k y σ_h para cada iteración. En la siguiente tabla 7.7 se recogen los resultados:

En cuanto a la precisión, el resultado de PROTEINS es algo más modesto, pero en el mejor de las iteraciones da resultados de hasta el $86,49\,\%$. En cambio, en MUTAG hay que quitar importancia a las iteraciones que consiguen una clasificación perfecta porque el conjunto de pruebas es pequeño y el conjunto de datos está desequilibrado. En la siguiente tabla 7.8 se pueden observar las medias obtenidas en las 10 iteraciones:

Iteración	PROTEINS	MUTAG
1	0.8214	0.8947
2	0.8571	0.8421
3	0.8739	0.8421
4	0.8198	0.8947
5	0.8198	0.9474
6	0.8108	1.0000
7	0.7297	0.9474
8	0.8378	0.8333
9	0.8649	0.8333
10	0.7387	0.8333

Tabla 7.7: Precisión por iteración con WKPI

Dataset	**Desviación Típica**	**Tiempo Total (minutos)**	**Precisión**
PROTEINS	0.0463	38.4146	0.8174
MUTAG	0.0573	9.1172	0.8868

Tabla 7.8: Resultados medios de WKPI.

La precisión obtenida con este modelo en ambos *datasets* es muy buena, ambas superan el 80 %. La desviación típica no es muy grande, ambas rondan el 0,05 % lo que indica que no hay mayor sensibilidad a la selección de los conjuntos de entrenamiento y prueba. Por último, hay que destacar el tiempo de ejecución porque es muy grande. Para una sola iteración se tardan 38 minutos para PROTEINS y 9 minutos para MUTAG. Además, en estos cálculos no se indica el tiempo empleado para generar las imágenes de persistencia, por lo que se puede convertir en una desventaja importante para aplicar el modelo. A diferencia de los datos en forma de grafos, no se ha calculado una función de pérdida debido a que no ha resultado de gran utilidad en las comparaciones entre modelos y podemos medir su desempeño de otras maneras. En esta tabla 7.9, se puede observar la precisión obtenida por grupos:

Dataset	Etiqueta	Precisión
PROTEINS	0	90,17 %
PROTEINS	1	69,10 %
MUTAG	0	95,81 %
MUTAG	1	72,73 %

Tabla 7.9: Precisión por grupos con el modelo WKPI

Aunque en ambos conjuntos de datos hay un grupo con mayor precisión, no existe un desbalanceo muy significativo, ya que la precisión del grupo con peor resultado es del 70 % más o menos.

7.2.2. Aplicación del modelo SVM en Imágenes de Persistencia

El segundo modelo escogido para imágenes de persistencia es el *SVM* o Máquina de Soporte Vectorial, un algoritmo de aprendizaje supervisado utilizado tanto para problemas de clasificación como de regresión. Fue desarrollado por Vladímir Vapnik en 1997 [Bar15]. El motivo de usar este modelo es que es apropiado para clasificar imágenes en dos grupos y tiene mejor rendimiento que los tradicionales esquemas de búsquedas de refinamiento. Sus limitaciones aparecen con imágenes de mucho tamaño y gran complejidad, cuestión que no debería ser problemática por tener imágenes unicanales de solo 400 píxeles.

En la siguiente tabla 7.10 se muestran los resultados obtenidos. En este caso, el conjunto de validación y prueba tienen un 10 % de tamaño sobre el *dataset* original y el de prueba otro 10 %, el resto de los datos son para entrenamiento. El SVM aprenderá los hiperparámetros adecuados en los datos de validación y, después, predecirá los valores del conjunto de testeo.

Semilla	**Precisión Validación**		**Precisión Prueba**	
	PROTEINS	**MUTAG**	**PROTEINS**	**MUTAG**
100	0.7027	0.8333	0.7411	0.75
507	0.6937	0.9444	0.8304	1.0
57	0.8198	0.8889	0.7411	0.85
101	0.7387	0.8333	0.7232	1.0
102	0.6757	0.8889	0.6964	0.8
103	0.6937	1.0	0.6875	0.95
104	0.7117	1.0	0.7411	0.9
105	0.7297	0.8333	0.6964	0.9
106	0.7658	0.8889	0.7679	0.95
107	0.7568	0.8889	0.7054	0.9

Tabla 7.10: 10 ejecuciones sobre imágenes de persistencia en PROTEINS y MUTAG con SVM

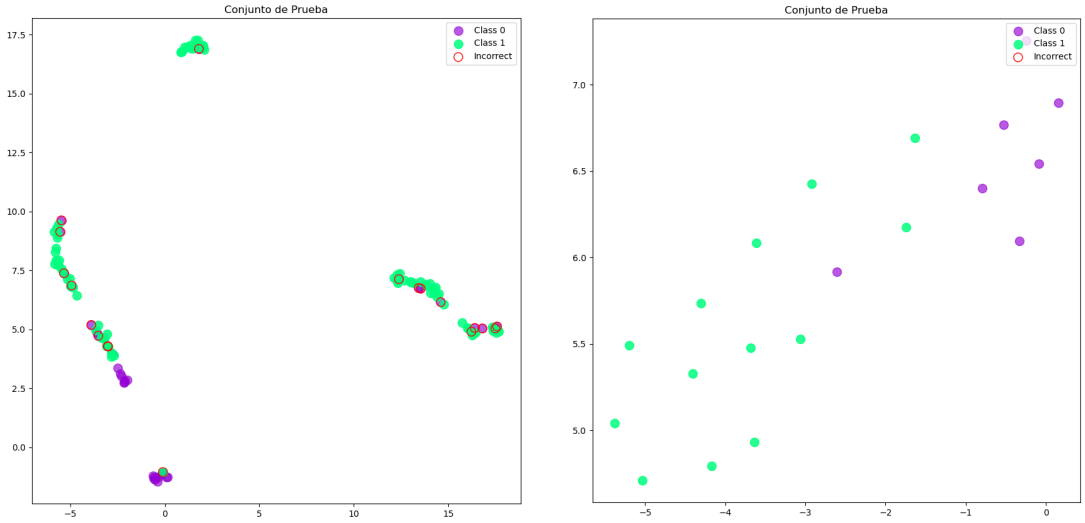
Se puede observar que hay variaciones de las precisiones de validación y prueba en las iteraciones, pero a la hora de hacer la media, se consigue la misma tasa de precisión en validación y testeo. Esta obtiene un valor del 73 % para el *dataset* PROTEINS y del 90 % para MUTAG. En un principio, el modelo para MUTAG se comporta bien, pero consigue unos resultados limitados para PROTEINS. En la tabla 7.11 se puede ver la media de la precisión, la media del tiempo y la desviación típica de la precisión del conjunto de prueba.

Para evaluar completamente el desempeño, se van a generar los diagramas UMAP 7.8 para visibilizar si se sufre de problemas de desbalanceo.

A primera vista, en el conjunto PROTEINS no se visualiza un desbalanceo importante, mientras que en el *dataset* MUTAG no se obtiene ningún error de

Dataset	Desviación Típica	Precisión	Tiempo Total (segundos)
PROTEINS	0.042829	0.73305	8.51
MUTAG	0.081650	0.9	4.29

Tabla 7.11: Resultados del modelo SVM en el conjunto de prueba



(a) PROTEINS (b) MUTAG

Figura 7.8: UMAP en el modelo SVM en conjunto de testeo

clasificación para esta selección de los datos de entrenamiento, prueba y validación. Por ello, en la siguiente tabla 7.12 se calcula la precisión por grupos de clasificación.

Dataset	Etiqueta	Precisión
PROTEINS	0	52.78 %
PROTEINS	1	97.37 %
MUTAG	0	100.00 %
MUTAG	1	100.00 %

Tabla 7.12: Precisión por grupos con el modelo SVM

El desbalanceo en PROTEINS es notable, ya que clasifica bien un grupo, pero el otro lo reparte indistintamente entre los dos clústeres. No se ha visto necesidad de hacer más cálculos del desbalanceo en MUTAG debido a la alta precisión y el bajo número de datos en el conjunto de prueba. Por otra parte, los tiempos de ejecución han sido muy reducidos en este caso, pudiendo hacer los cálculos en unos pocos segundos. En cuanto a la desviación típica, el *dataset* PROTEINS se ha visto dentro de unos niveles habituales para los modelos que se están evaluando. Sin embargo, el *dataset* MUTAG tiene una σ bastante grande, pero habitual cuando se tiene un tamaño reducido en el conjunto de pruebas.

7.2.3. Comparativa entre WKPI y SVM

En esta sección, compararemos el desempeño de los modelos para el descriptor molecular imagen de persistencia. En la siguiente tabla 7.13 se pueden ver los resultados reunidos.

Dataset	Desviación Típica		Precisión		Tiempo Total		Desbalanceo	
	WKPI	**SVM**	**WKPI**	**SVM**	**WKPI (min)**	**SVM (seg)**	**WKPI**	**SVM**
PROTEINS	0.0463	0.042829	0.8174	0.73305	38.4146	8.51	21.07 %	44.59 %
MUTAG	0.0573	0.081650	0.8868	0.9	9.1172	4.29	23.08 %	30.66 %

Tabla 7.13: Comparación de los resultados de los modelos WKPI y SVM

En primer lugar, la desviación típica no ha variado mucho para el *dataset* PROTEINS en los dos modelos analizados. Al contrario, MUTAG ha sufrido un incremento considerable de la desviación estándar en el modelo SVM. Esto puede ser debido a un problema de sobreajuste con SVM en MUTAG, el modelo podría ajustarse demasiado a unas características de los conjuntos de entrenamiento que no estuvieran presentes en todos los conjuntos de prueba.

En segundo lugar, no se pueden extraer conclusiones certeras sobre qué modelo obtiene mejor precisión. Para el *dataset* PROTEINS, donde hay mayor número de elementos, los resultados son considerablemente mejores en WKPI. Por otra parte, en MUTAG ambos modelos obtienen rendimientos parecidos. Se podría argumentar que WKPI sería más fiable, ya que obtiene precisiones mejores o iguales con una menor desviación típica.

En tercer lugar, la ventaja, en cuanto al tiempo de computación, de SVM frente a WKPI es clara. Los tiempos son muy superiores con WKPI, aunque podría no ser un factor determinante si se dispusiese de mejores ordenadores o el tiempo no supusiera demasiado inconveniente.

Para finalizar se van a evaluar los desbalanceos. Se puede ver de forma clara que WKPI tiene menor desbalanceo que SVM. Esto significa que SVM evalúa bien una clase, pero no tan bien la otra. Podría resultar en una desventaja importante porque podría caer en el problema del clasificador "trivial", es decir, agrupar las dos clases en un único grupo y dar una engañosa alta tasa de precisión porque los datos de entrada están desequilibrados.

7.3. Aplicación de modelos de Aprendizaje Automático en SMILES

Para terminar con la comparativa, se ha realizado una clasificación partiendo del descriptor molecular SMILES. Para esta ocasión, solo se disponían de los *datasets* MUTAG y Tox21, cabe recalcar que estos están bastante desequilibrados teniendo una proporción de 66,5 % y 88,3 % de elementos en sus clases mayoritarias respectivamente.

Se han realizado estudios sobre clasificar moléculas con este descriptor [Hir+18]. Sin embargo, no es el descriptor más popular para esta tarea. En este caso, se ha decidido recurrir a una Red Neuronal Recurrente (RNN) por su utilidad en procesamiento del lenguaje natural. Aunque los *strings* de los SMILES no se pueden considerar como lenguaje natural, su información previa puede generar un contexto que sea beneficioso en la predicción y puede operar con datos de entrada de longitud desigual. Además, se ha utilizado exitosamente con anterioridad [Arú+19].

Para valorar los resultados se ha creado una tabla 7.14 donde ver exactamente cómo se ha clasificado las moléculas en el mejor de las 10 iteraciones. La razón de ello es que ambos *datasets* están desequilibrados y, por tanto, unas tasas de precisión altas podrían ser engañosas sin el debido contexto.

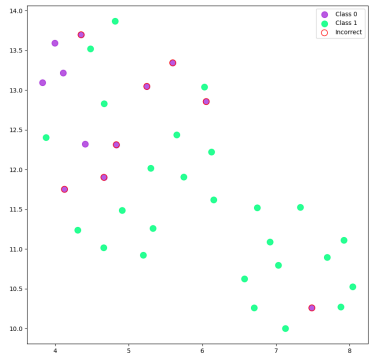
	MUTAG	**Tox21**
Verdadero positivo	26	1432
Falso positivo	8	201
Verdadero negativo	4	1
Falso negativo	0	0

Tabla 7.14: Resultados del modelo RNN con SMILES

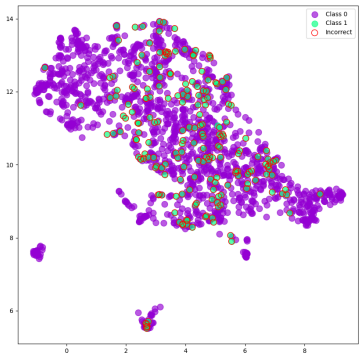
Como se puede observar, en los dos *datasets* la mayoría de los elementos están clasificados en los grupos de "positivo". Esto significa que el modelo tiene un sesgo importante y mete a casi todos los elementos en un único grupo. Además, como los datos están desequilibrados, es decir, hay más de un tipo que de otro, la precisión que se obtiene es alta, pero sin que sea un índice representativo. A pesar de todo, este sesgo no es tan significativo en el *dataset* MUTAG, pero incluso aquí clasifica incorrectamente al 66,66 % de los elementos de la clase minoritaria. En las siguientes imágenes 7.9 se puede comprobar el problema del sesgo visualmente:

La siguiente tabla 7.15 contiene la información para discutir brevemente el desempeño medio de las 10 iteraciones que se han probado.

Como se puede observar, la precisión es relativamente alta, pero tenemos que tener en cuenta el desbalanceo de los datos. Por otra parte, la desviación estándar es mucho más alta en MUTAG probablemente por el reducido tamaño y por un posible sobreajuste del modelo. En cuanto al tiempo, se considera que es adecuado para el tamaño que suelen tener los *dataset* de moléculas de clasificación.



(a) MUTAG



(b) Tox 21

Figura 7.9: UMAP sobre los resultados del modelo RNN con SMILES

Dataset	**Desviación Típica**	**Precisión**	**Tiempo Total (segundos)**
Tox21	0.0139	0.8505	112.13
MUTAG	0.0477	0.7694	39.80

Tabla 7.15: Resultados del modelo RNN en el conjunto de prueba

7.4. Comparativa general

En esta sección se van a comparar los resultados de los cinco modelos escogidos para los tres descriptores.

Descriptor	**Modelo**	***Dataset***	**Desviacion Tipica**	**Precisión**	**Tiempo Total** (seg)
Grafos	HGP-SL	PROTEINS	0.0494	0.7536	673.75
		MUTAG	0.0224	0.8000	23.06
		NCI1	0.0087	0.7579	942.71
		Tox21	0.0027	0.8987	1125.06
	GCNConv	PROTEINS	0.0103	0.7321	292.507
		MUTAG	0	0.7000	32.568
		NCI1	0.0080	0.6591	799.578
		Tox21	0	0.8814	1438.936
Imágenes	SVM	PROTEINS	0.0428	0.7331	8.51
		MUTAG	0.0817	0.9000	4.29
	WKPI	PROTEINS	0.0463	0.8174	38.415 (min)
		MUTAG	0.0573	0.8868	9.117 (min)
SMILES	RNN	MUTAG	0.0477	0.7694	39.80
		Tox21	0.0139	0.8505	112.13

Tabla 7.16: Tabla de resultados global

Cabe recalcar que se han realizado 10 iteraciones por cada modelo en cada *dataset*. No ha sido posible seleccionar los mismos conjuntos de entrenamiento, validación y prueba porque no se tenía la correspondencia de las moléculas entre distintos *datasets* y porque algunos no han requerido de un conjunto de validación

explícito. Aun así, por el método de validación cruzada escogido, los datos de la tabla 7.16 son perfectamente comparables.

En primer lugar, se discutirá la desviación típica. El aspecto más relevante ha sido el modelo escogido, ya que los mismos *datasets* han obtenido desiguales resultados. Además, el descriptor molecular grafos ha obtenido una menor desviación típica que las demás, siendo el modelo GCNConv el menos sensible a los datos de entrada. Por el otro lado, el modelo SVM, que se nutre con imágenes de persistencia, ha obtenido los peores resultados. En líneas generales, se han obtenido resultados aceptables para la desviación estándar.

En próximo lugar se va a valorar el tiempo de computación. Con diferencia, el modelo que más ha tardado ha sido el WKPI y el que menos el SVM, por lo que se ve claramente que el factor determinante no es el tipo de descriptor escogido, sino el modelo. Naturalmente, cuanto mayor sea el *dataset* más tardará el modelo. En general, los tiempos no han sido muy grandes teniendo en cuenta que se ha utilizado un ordenador convencional.

Para evaluar el parámetro más importante, la precisión, es necesario tener en cuenta las consideraciones que se han hecho en este capítulo, sobre todo, los referentes al desbalanceo de datos. De forma resumida, se han recogido los datos en la siguiente imagen: 7.10.

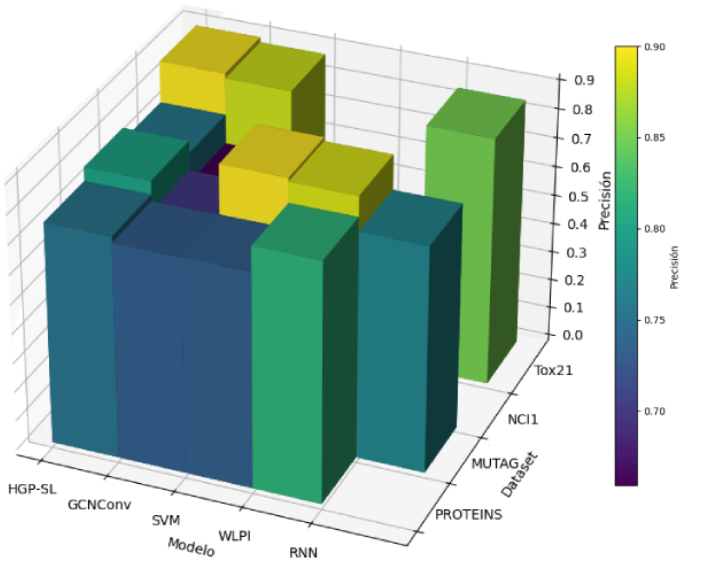


Figura 7.10: Precisión por *datasets* y modelos

Como se puede observar, los grafos han tenido mejor desempeño solo en el *dataset* Tox21. En el resto de los casos, los otros descriptores han obtenido mejor resultado. Tomando como referencia el *dataset* MUTAG porque es de especial interés al haber sido entrenado con los cinco modelos distintos, se puede observar

que las imágenes de persistencia han dado los mejores resultados. Por otro lado, la RNN con SMILES ha obtenido peores resultados que otros modelos.

Otro aspecto interesante es el desempeño de cada modelo en los distintos conjuntos de datos. Se pueden observar diferencias muy notorias en precisión como, por ejemplo, en el modelo GCNConv. En este caso se tiene un claro problema de generalización. Sin embargo, con otros modelos no queda muy clara la cuestión porque no se han entrenado más que en dos conjuntos de datos y es difícil extraer conclusiones.

Capítulo 8

Conclusiones

La principal conclusión de este trabajo es que los modelos de aprendizaje automático son solventes para realizar tareas de clasificación de moléculas. Los resultados obtenidos han ido en consonancia con los de la literatura científica. Por tanto, su potencial para aplicarlos en escenarios reales queda demostrado y se pueden utilizar en sustitución a los métodos alternativos que existían hasta el momento.

Otro de los objetivos principales de este trabajo era evaluar la relevancia de la correcta preparación y preprocesado de los datos para el éxito de estos modelos. Habiendo analizado los datos obtenidos se puede concluir que este es un facto crucial. La razón principal radica en la necesidad que tienen la mayoría de modelos de recibir información vectorizada. Precisamente, la técnica con mejores resultados, la WKPI, requiere una meticulosa conversión de grafos en imágenes de persistencia, incluyendo la información determinante para el clasificado. Por otro lado, existen descriptores, como los SMILES, que, aunque puedan valer para otras tareas, no presentan un buen desempeño en tareas de clasificación, tal y como se ha comprobado en las anteriores secciones, por lo que un preprocesado exitoso tiene que estar orientado para que se utilice en clasificación después.

Una contrapartida de los modelos aplicados es que no todos han generalizado bien a los distintos conjuntos de datos. Esto podría ser debido a la dificultad de detectar ciertas características, es decir, algunas propiedades podrían resultar más distinguibles para la clasificación que otras. Sin embargo, esta variabilidad en el desempeño no disminuye el potencial de los métodos de aprendizaje automático para la clasificación molecular. Al contrario, destaca la importancia de la investigación para desarrollar y mejorar algoritmos que sean capaces de abordar una variedad aún mayor de problemas y *datasets* con eficacia.

Otra desventaja es el desbalanceo de las técnicas. En algunos modelos ha sido muy significativo. Esto se puede deber al desequilibrio de los datos de entrada, por lo que el modelo encontraría dificultades para aprender sobre la clase infrarrepresentada. Sin embargo, también se puede deber a las limitaciones de los propios modelos. Después de hacer el estudio comparativo, se concluye que el modelo ha tenido mayor peso. Como se puede observar en la tabla 7.13, en el modelo WKPI

el desbalanceo ha sido parecido en PROTEINS y en MUTAG cuyas clases minoritarias representan el 41 % y el 33,33 % respectivamente, mientras que con SVM los desbalanceos han sido del 44,59 % y el 30,66 % respectivamente. Por lo que existen modelos capaces de mitigar esas infrarrepresentaciones.

Otro aprendizaje que se ha obtenido con este trabajo ha sido que los modelos diseñados ex profeso para la clasificación molecular han obtenido mejores resultados que los modelos más tradicionales. No solo han mejorado la precisión, sino que han generalizado mejor y con menores tasas de desbalanceo. Para este punto se ha tenido en cuenta el desempeño de WKPI frente a SVM y HGP-SL frente a GCNConv. Los modelos WKPI y HGP-SL, que han sido especialmente diseñados para esta tarea, han sido notablemente mejores. Esto enfatiza la necesidad de investigar en este campo tan concreto en lugar de recurrir a soluciones más generalistas.

Antes de concluir, se quiere exponer una vía que se ha intentado para mejorar los resultados obtenidos con los modelos. Esta consistía en aplicar el método de reducción de dimensionalidad UMAP para facilitar la tarea a los algoritmos de aprendizaje. Es una vía que se ha recorrido con éxito en otros campos [AKC20]. Sin embargo, los *datasets* han resultado ser demasiado complicados como para poder aplicarlo. En la imagen 8.1, se puede observar el conjunto PROTEINS reducido a dos dimensiones. Como se puede apreciar, el clúster en forma de línea morada que se encuentra en el centro solo contiene elementos de un grupo, es decir, UMAP serviría con esos puntos para poder clasificarlos. Sin embargo, como las otras dos líneas que se encuentra a los lados contienen indistintamente puntos verdes y morados, no se ha podido llevar a cabo la clusterización con UMAP. Esto refuerza la vía de la clasificación molecular mediante algoritmos de aprendizaje automático.

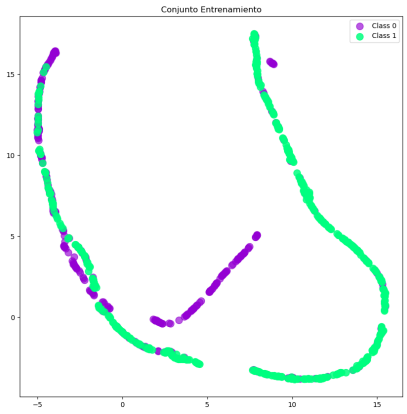


Figura 8.1: UMAP en el modelo WKPI sobre el conjunto de entrenamiento de PROTEINS

Por último, se observa que la efectividad de los modelos de aprendizaje au-

tomático puede variar significativamente dependiendo de la estructura y características intrínsecas de cada conjunto de datos. Por ejemplo, algunos conjuntos de datos pueden presentar más ruido, desbalanceos de clase, o estructuras más complejas que complican el proceso de clasificación. Por eso es tan importante cuidar el diseño del modelo para los conjuntos de datos en concreto. Este estudio ha puesto de manifiesto que los modelos de aprendizaje automático son herramientas versátiles, pero su eficacia en una tarea en particular puede ser altamente dependiente de cómo se aborden estos desafíos únicos propios de los *datasets* y de los modelos de aprendizaje seleccionados.

Bibliografía

[Abu+18] Sami Abu-El-Haija et al. «N-GCN: Multi-scale Graph Convolution for Semi-supervised Node Classification». En: (feb. de 2018).

[Adv14] National Center for Advancing Translational Sciences. *Tox 21 Challenge*. https://tripod.nih.gov/tox21/challenge/. 2014.

[AI18] Meta AI. *Graph Classification on PROTEINS*. 2018. URL: https:%20//paperswithcode.com/sota/graph-classification-on-proteins.

[AKC20] Mebarka Allaoui, Mohammed Lamine Kherfi y Abdelhakim Cheriet. «Considerably Improving Clustering Algorithms Using UMAP Dimensionality Reduction Technique: A Comparative Study». En: *Image and Signal Processing*. Ed. por Abderrahim El Moataz et al. Cham: Springer International Publishing, 2020, págs. 317-325. ISBN: 978-3-030-51935-3.

[Arú+19] Josep Arús-Pous et al. «Randomized SMILES strings improve the quality of molecular generative models». En: *Journal of Cheminformatics* 11.1 (2019), pág. 71. ISSN: 1758-2946. DOI: 10.1186/s13321-019-0393-0. URL: https://doi.org/10.1186/s13321-019-0393-0.

[Bar15] Lauren Barghout. «Spatial-Taxon Information Granules as Used in Iterative Fuzzy-Decision-Making for Image Segmentation». En: *Granular Computing and Decision-Making. Studies in Big Data*. Vol. 10. Archived from the original (PDF) on 2018-01-08. Retrieved 2018-01-08. 2015, págs. 285-318. ISBN: 978-3-319-16828-9. DOI: 10.1007/978-3-319-16829-6_12. URL: https://link.springer.com/chapter/10.1007%2F978-3-319-16829-6_12.

[Bor+05] Karsten M. Borgwardt et al. «Protein function prediction via graph kernels». En: *Proceedings of the 2005 International Conference on Bioinformatics*. Vol. 21. Bioinformatics supplement 1. Jun. de 2005, págs. 47-56.

[Gil+17] Justin Gilmer et al. «Neural Message Passing for Quantum Chemistry». En: *Proceedings of the 34th International Conference on Machine Learning*. Ed. por Doina Precup y Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, ago. de 2017,

págs. 1263-1272. URL: https://proceedings.mlr.press/v70/gilmer17a.html.

[God18] Primož Godec. «Graph Embeddings — The Summary». En: *Towards Data Science* (2018). https://towardsdatascience.com/graph-embeddings-the-summary-cc6075aba007.

[Gol+99] T. R. Golub et al. «Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring». En: *Science* 286.5439 (1999), págs. 531-537.

[Ham20] William L. Hamilton. *Graph Representation Learning*. Vol. 14. Synthesis Lectures on Artificial Intelligence and Machine Learning 3. 2020, págs. 12-17.

[Hir+18] Maya Hirohara et al. «Convolutional neural network based on SMILES representation of compounds for detecting chemical motif». En: *BMC Bioinformatics* 19.19 (2018), pág. 526. ISSN: 1471-2105. DOI: 10.1186/s12859-018-2523-5. URL: https://doi.org/10.1186/s12859-018-2523-5.

[JLC16] Stanisław Jastrzębski, Damian Leśniak y Wojciech Czarnecki. «Learning to SMILE(S)». En: (feb. de 2016).

[Kip16] Thomas Kipf. *Graph Convolutional Networks*. Blog post. Sep. de 2016. URL: https://tkipf.github.io/graph-convolutional-networks/.

[KW16] Thomas Kipf y Max Welling. «Semi-Supervised Classification with Graph Convolutional Networks». En: (sep. de 2016).

[Mar22] Dominic D. Martinelli. «Generative machine learning for de novo drug discovery: A systematic review». En: *Computers in Biology and Medicine* 145 (2022), pág. 105403. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2022.105403. URL: https://www.sciencedirect.com/science/article/pii/S0010482522001950.

[McI+18] Leland McInnes et al. «UMAP: Uniform Manifold Approximation and Projection». En: *Journal of Open Source Software* 3 (sep. de 2018), pág. 861. DOI: 10.21105/joss.00861.

[NGL05] Greeshma Neglur, Robert L. Grossman y Bing Liu. «Assigning Unique Keys to Chemical Compounds for Data Integration: Some Interesting Counter Examples». En: *Data Integration in the Life Sciences*. Ed. por Bertram Ludäscher y Louiqa Raschid. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, págs. 145-157. ISBN: 978-3-540-31879-8.

[Qin+22] Kun Qin et al. «Using graph convolutional network to characterize individuals with major depressive disorder across multiple imaging sites». En: *EBioMedicine* 78 (2022), pág. 103977. DOI: 10.1016/j.ebiom.2022.103977.

[Ran96] Milan Randić. «Molecular bonding profiles». En: Berlin, Heidelberg: Journal of Mathematical Chemistry, 1996.

[Wei88] David Weininger. «SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules». En: *Journal of Chemical Information and Computer Sciences* 28.1 (1988), págs. 31-36. DOI: 10.1021/ci00057a005. eprint: https://doi.org/10.1021/ci00057a005. URL: https://doi.org/10.1021/ci00057a005.

[WH21] Jiande Wu y Chindo Hicks. «Breast Cancer Type Classification Using Machine Learning». En: *Journal of Personalized Medicine* 11.2 (2021). ISSN: 2075-4426. DOI: 10.3390/jpm11020061. URL: https://www.mdpi.com/2075-4426/11/2/61.

[WK06] Nikil Wale y George Karypis. «Comparison of Descriptor Spaces for Chemical Compound Retrieval and Classification». En: *Sixth International Conference on Data Mining (ICDM'06)*. 2006, págs. 678-689. DOI: 10.1109/ICDM.2006.39.

[WWK08] Nikil Wale, Ian A. Watson y George Karypis. «Comparison of Descriptor Spaces for Chemical Compound Retrieval and Classification». En: *Knowledge and Information Sys.* 14.3 (2008), págs. 347-375. ISSN: 0219-3116. DOI: 10.1007/s10115-007-0103-5. URL: https://doi.org/10.1007/s10115-007-0103-5.

[WWW89] David Weininger, Arthur Weininger y Joseph L. Weininger. «SMILES. 2. Algorithm for generation of unique SMILES notation». En: *Journal of Chemical Information and Computer Sciences* 29.2 (1989), págs. 97-101. DOI: 10.1021/ci00062a008. eprint: https://doi.org/10.1021/ci00062a008. URL: https://doi.org/10.1021/ci00062a008.

[Zha+19a] Zhen Zhang et al. «Hierarchical Graph Pooling with Structure Learning». En: *arXiv preprint arXiv:1911.05954* (2019). URL: https://github.com/cszhangzhen/HGP-SL.

[Zha+19b] Zhen Zhang et al. «Hierarchical Graph Pooling with Structure Learning». En: *CoRR* abs/1911.05954 (2019). arXiv: 1911.05954. URL: http://arxiv.org/abs/1911.05954.

[ZW19] Qi Zhao y Yusu Wang. «Learning metrics for persistence-based summaries and applications for graph classification». En: *Advances in Neural Information Processing Systems*. Ed. por H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/12780ea688a71dabc%20284b064add459a4-Paper.pdf.