

## Impacto de Factores Topológicos y de Desbalance en la Clasificación de Nodos con GCNs

Tatiana S. Parlanti<sup>1</sup>, Carlos A. Catania<sup>1</sup>, and Luis G. Moyano<sup>2</sup>

<sup>1</sup> Universidad Nacional de Cuyo, Facultad de Ingeniería, Laboratorio de Sistemas Inteligentes (LABSIN), Mendoza, Argentina

<sup>2</sup> División de Física Estadística e Interdisciplinaria, Centro Atómico Bariloche, Bariloche, Argentina

{tatiana.parlanti,harpo}@ingenieria.uncuyo.edu.ar

**Resumen** Las Redes Neuronales Convolucionales para Grafos (GCNs) han demostrado ser sumamente efectivas en la resolución de problemas relacionados con grafos, ya que no solo consideran las características individuales de los nodos, sino que también capturan la esencia topológica del grafo. No obstante, la escasez de conjuntos de datos públicos representa un obstáculo en la evaluación y comparación objetiva de estas redes en diversos contextos. En este artículo, se abordan las limitaciones inherentes de las GCNs, enfocándose específicamente en el impacto de los aspectos topológicos y el desbalance de clases en la tarea de clasificación de nodos. A través del algoritmo *Contextual Stochastic Block Model* (CSBM) se generan distintos grafos sintéticos que varían en características y topologías. De esta manera, se realiza una exploración exhaustiva de las capacidades de las GCNs en distintos escenarios. Los hallazgos iniciales remarcan la importancia fundamental de las características de los nodos en la clasificación y ponen de manifiesto los desafíos que surgen al tratar con escenarios de desbalance de clases.

**Keywords:** Redes Complejas, Redes Neuronales para Grafos, Grafos Sintéticos

### 1. Introducción

Un grafo  $G = (V, E)$  es un conjunto de nodos o vértices  $V$  y un conjunto de enlaces o aristas  $E$  entre dichos nodos. El interés de estudiar estas estructuras es que muchos problemas se pueden interpretar a la luz de un grafo, describiendo apropiadamente quiénes son los nodos y qué representan las aristas que los unen. Así es que en los últimos años, las redes neuronales para grafos, conocidas como GNN por sus siglas en inglés (*Graph Neural Networks*), han tenido un marcado auge debido a su aplicación en numerosos problemas que involucran grafos, que van desde clasificación de proteínas según la estructura de la misma, hasta recomendación de contenido a usuarios en una plataforma, o predicción del tema del cual trata un paper teniendo en cuenta los trabajos que este cita y los que lo citan [3].

Uno de los primeros trabajos que exploró las redes neuronales para grafos fue el de Kipf y Welling [4], donde presentaron las redes convolucionales para grafos o GCN (*Graph Convolutional Networks*). Este trabajo tuvo gran impacto porque mostró que para un conjunto de nodos no etiquetados, se puede predecir su etiqueta usando no solamente las características de cada nodo, sino también -y allí radica lo novedoso- la información propia de las conexiones entre nodos. Es decir, el grafo subyacente que

describe el problema guarda información valiosa a la hora de entrenar y predecir usando una red neuronal.

Tomando como referencia el paper de Kipf y Welling [4], en este trabajo se tiene por objetivo explorar las GCNs a fin de determinar sus alcances y limitaciones. Para esto, se pondrá a prueba una GCN de dos capas, en diferentes conjuntos de grafos sintéticos, para clasificar los nodos de los mismos en un problema de clasificación binaria.

Lo que sigue está organizado de la siguiente forma: se comienza por una descripción teórica de las GNN, para seguir con una pequeña descripción del hardware y software utilizados, así como de los experimentos realizados. Luego se exponen los resultados y discusiones y finalmente las conclusiones y trabajo a futuro.

## 2. Breve descripción teórica de las GNNs

Antes de exponer los experimentos a realizar, se muestra a continuación una breve descripción teórica de las GNNs, que comienza por la notación a utilizar en todo el trabajo.

### 2.1 Notación

Dado un grafo  $G = (V, E)$ , se presentan a continuación varias definiciones relacionadas al concepto de grafo:

- Dos nodos  $u, v \in V$  están conectados si  $(u, v) \in E$ .
- Dado un vértice  $u \in V$ , se define el entorno de  $u$  como  $N(u) = \{v \in V \mid (u, v) \in E\}$ .
- Un grafo se dice no dirigidos si  $(u, v) \in E \Leftrightarrow (v, u) \in E$ .
- Un grafo se dice dirigido si  $(u, v) \in E$  no implica necesariamente que exista la conexión inversa. En este caso, el par  $(u, v)$  representa el enlace desde  $u$  hasta  $v$ , que gráficamente se muestra con una flecha de  $u$  a  $v$ . ■ Una manera de representar a un grafo es a partir de su matriz de adyacencia  $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$ , que se define como  $A_{ij} = 1$  si  $(i, j) \in E$ , y  $A_{ij} = 0$  caso contrario. Nótese que si el grafo es no dirigido, entonces la matriz de adyacencia es simétrica. Por el contrario, si el grafo es dirigido, la matriz de adyacencia no es simétrica.
- Dado un grafo, si cada uno de sus nodos tiene  $d$  características, se define la matriz de características (*features*) como  $\mathbf{X} \in \mathbb{R}^{|V| \times d}$ , donde la  $i$ -ésima fila de  $\mathbf{X}$  contiene las  $d$  características del nodo  $i \in V$ .

### 2.2 Incrustaciones y mensajes neuronales

La característica que define a una GNN es que utiliza una forma de “paso de mensajes neuronales” en la que vectores de mensajes se intercambian entre nodos y se actualizan mediante redes neuronales. Para esto se generan incrustaciones o *embeddings* ocultos para cada nodo  $u \in V$ , a partir del grafo  $G = (V, E)$  y la matriz de características de sus nodos  $\mathbf{X}$  [3].

Así, durante cada paso de mensajes en una GNN, cada *embedding* oculto  $\mathbf{h}_u^{(k)}$  correspondiente a cada nodo  $u \in V$  es actualizado teniendo en cuenta la información acumulada a partir de los vecinos de  $u$ , es decir de su entorno  $N(u)$ . Dicha actualización está dada por:

$$\begin{aligned}\mathbf{h}_u^{(k+1)} &= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}\left(\{\mathbf{h}_v^{(k)} \mid v \in N(u)\}\right)\right) \\ &= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{N(u)}^{(k)}\right),\end{aligned}$$

donde UPDATE y AGGREGATE son funciones diferenciables,  $\mathbf{m}_{N(u)}$  es el “mensaje” que se recopila de los vecinos de  $u$ , y  $k$  indica la iteración o capa de la GNN (que para  $k = 0$  se tiene que  $\mathbf{h}_u^{(0)} = x_u$ , es decir que se utilizan las características de cada nodo). Es decir, en cada iteración la incrustación se actualiza teniendo en cuenta el mensaje recibido por parte de su entorno y la incrustación en la capa anterior. Nótese además que, dado que la función AGGREGATE toma un conjunto como argumento, una GNN definida de esta manera es equivariante a permutaciones por diseño [3].

En el caso particular de las redes GCN, las funciones UPDATE y AGGREGATE son sumas. Además, se aplica una normalización diferente dependiendo de si se trata de un grafo dirigido o uno no dirigido:

$$\mathbf{m}_{N(u)} = \begin{cases} \sum_{v \in N(u) \cup \{u\}} \frac{\mathbf{h}_v}{\sqrt{|N(u)||N(v)|}} & \text{si el grafo es no dirigido} \\ \sum_{v \in N(u) \cup \{u\}} \frac{\mathbf{h}_v}{|N(u)|} & \text{si el grafo es dirigido} \end{cases}$$

Así, una GCN define su actualización como:

$$\mathbf{h}_u^{(k+1)} = \sigma\left(\mathbf{W}^{(k)} \mathbf{m}_{N(u)}^{(k)}\right), \quad \forall u \in V, \quad (1)$$

donde  $\mathbf{W}$  es una matriz de pesos (parámetros) que aprende la red, y  $\sigma$  es una función no lineal, por lo general  $\sigma = \text{ReLU}$ .

Por otro lado, aprovechando la notación matricial y sus ventajas a la hora de programar, se puede redefinir la ecuación 1 para actualizar todos los vértices a la vez:

$$\mathbf{H}^{(k+1)} = \begin{cases} \sigma\left(\bar{\mathbf{D}}^{-1/2} \bar{\mathbf{A}} \bar{\mathbf{D}}^{-1/2} \mathbf{H}^{(k)} \mathbf{W}^{(k)}\right) & \text{si el grafo es no dirigido} \\ \sigma\left(\bar{\mathbf{D}}^{-1} \bar{\mathbf{A}} \mathbf{H}^{(k)} \mathbf{W}^{(k)}\right) & \text{si el grafo es dirigido} \end{cases}, \quad (2)$$

donde  $\bar{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ , con  $\mathbf{A}$  la matriz de adyacencia e  $\mathbf{I}$  la matriz identidad, y donde  $\bar{\mathbf{D}}$  es la matriz (diagonal) de grados dada por  $\bar{D}_{ii} = \sum_j \bar{A}_{ij}$ .

### 2.3 Hipótesis a poner a prueba

Teniendo en cuenta lo anterior, se tienen las siguientes hipótesis que se pondrán a prueba:

1. La GCN puede aprender a discriminar entre clases cuando están balanceadas y cuando existe una buena “separación” entre los nodos pertenecientes a cada una.
2. Si la topología del grafo es tal que las clases están lo suficientemente “separadas”, entonces la GCN puede aprender a discriminar entre clases independientemente del valor de las características de los nodos.
3. El aprendizaje de la GCN se puede ver afectado por el grado de desbalance entre clases.

### 3. Materiales y métodos

En esta sección se describen el hardware y software utilizado, y el detalle de los experimentos.

#### 3.1 Descripción de hardware y software utilizado

Para llevar a cabo los experimentos aquí planteados, se utilizó un procesador AMD FX(tm)-6300 Six-Core con sistema operativo Ubuntu 20.04.6 LTS, disponible en la Facultad de Ingeniería de la Universidad Nacional de Cuyo, que cuenta con una placa de video NVIDIA GeForce GTX 1080 Ti de 12 GB.

Usando la versión 3.8.10 de python, se utilizaron las siguientes bibliotecas principales: (a) `tensorflow`: versión 2.9.3; (b) `spektral` [2]: biblioteca para el aprendizaje profundo de grafos, basada en la API Keras y TensorFlow 2. Versión 1.2.0; (c) `pandas`: versión 1.5.3; (d) `numpy`: versión 1.23.4; (e) `matplotlib`: versión 3.6.2.

#### 3.2 Descripción de los experimentos realizados

Para comenzar, las bases de datos que se utilizaron en cada prueba fueron construidas para tal fin, es decir que se crearon grafos sintéticos modificando alguna particularidad específica dependiendo del experimento. En esta sección se describe el proceso de armado de grafos sintéticos, así como el pre-procesamiento llevado a cabo sobre los datos y la estructura de la red, y las métricas utilizadas para evaluar el desempeño de la GCN.

##### 3.2.1 Grafos sintéticos

Para llevar a cabo cada una de las pruebas, en primer lugar, se crearon diferentes conjuntos de grafos dirigidos sintéticos, siguiendo lo propuesto en [1], esto es usando el algoritmo *Contextual Stochastic Block Model* (CSBM), que genera tanto características para los nodos así como el grafo. Cada dataset creado consta de diez

grafos, cada uno de los cuales tiene 100 nodos<sup>1</sup>. Para poder construir dichos grafos es necesario definir:

- Número de clases: 2 (dos), por tratarse de un problema de clasificación binaria. A cada nodo se le asigna una etiqueta que lo identifica con la clase correspondiente.
- Número de características que tendrán los nodos: 2 (dos). Se eligió esta cantidad por simplicidad.
- Valor medio de las características por clase.
- Probabilidad de existencia de enlaces entre nodos de diferentes clases.

A la hora de definir los diferentes conjuntos de grafos, se modificaron la media de las características por clase, y la probabilidad de conexión entre los nodos, como muestra la Tabla 1, consiguiendo así diferentes “niveles de separación” entre las clases. Por un lado, con los tres tipos de separación, se armaron datasets donde las clases estaban balanceadas entre sí. Por otro lado, únicamente con la configuración de los casos “separados” y “mezclados” se crearon grafos donde las clases no estaban balanceadas, y la clase minoritaria representaba los siguientes porcentajes sobre el total: 1,2,5,10,20. Algunos ejemplos de estos grafos sintéticos se observan en las Figuras 1 y 2.

**Tabla 1.** Niveles de separación entre los nodos pertenecientes a diferentes clases.

Caso	Probabilidad de conexión	Media de características por clase
“separados”	0,8 0,2	$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$
	0,3 0,7	$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$
“juntos”	0,5 0,5	$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$
	0,5 0,5	$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$
“mezclados”	0,5 0,5	$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$
	0,5 0,5	$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

A su vez, en los casos de clases balanceadas se puso a prueba el comportamiento de la GCN al aplanar todas las *features* de los nodos a un valor constante, en este caso, 1 (uno). Estas pruebas tienen la particularidad de que los grafos sintéticos generados son iguales, desde un punto de vista topológico, a los generados anteriormente, es decir, los nodos guardan la misma proporción entre clases, presentan el mismo “nivel de separación” y tienen las mismas conexiones. La única diferencia es que al leer la información de las características, en lugar de las *features* generadas, asigna a cada nodo un mismo valor.

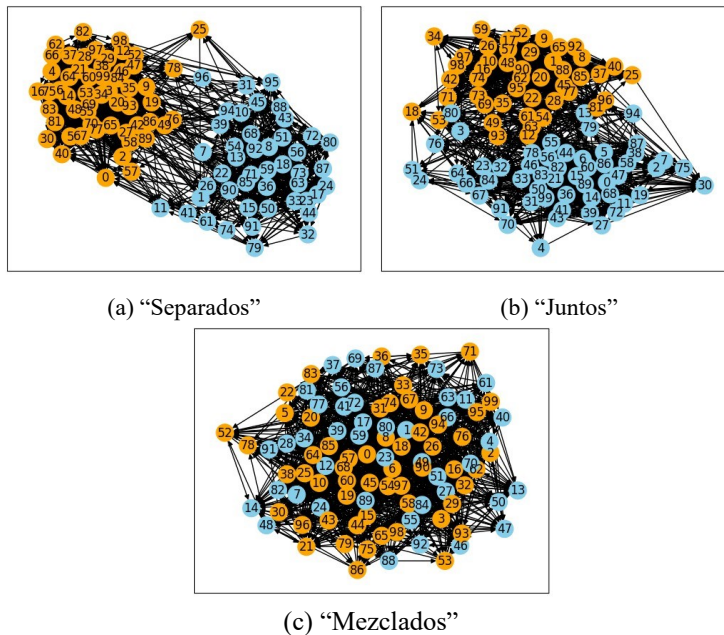
Una vez que se cargan los grafos en memoria, y antes de realizar cada prueba, se realiza un pre-procesamiento a los datos que está descrito en la próxima sección.

<sup>1</sup> La elección en el número de nodos radica en minimizar los tiempos de entrenamiento de la GCN.

### 3.2.2 Pre-procesamiento y estructura de la red

Cada prueba cuenta con su propio conjunto de grafos como datos de entrada. A su vez, por cada grafo se tienen tres “componentes”: la matriz de adyacencia, la matriz de características de los nodos y el vector de etiquetas de los nodos, usualmente representado como una matriz one-hot.

Antes de ingresar los datos a la red neuronal, se realiza un pre-procesamiento a la matriz de adyacencia  $A$  y a la matriz de características de los nodos  $X$ . Por un lado, como se explica en la sección 2.2, se calcula  $\bar{A} = A + I$ , donde  $I$  es



**Figura1.** Ejemplos de grafos con clases balanceadas.

la matriz identidad, para luego normalizar  $\bar{A}$ . Dicha normalización depende de si la matriz de adyacencia es o no simétrica. En este caso, como los grafos son dirigidos, siguiendo la ecuación 2, la normalización se calcula como  $D^{-1} \bar{A}$ . Por otro lado, la matriz de características se normaliza por filas, es decir, para cada fila de  $X$  se divide cada entrada por la suma de los elementos de dicha fila.

Durante el entrenamiento se usó una tasa de aprendizaje (*learning rate*) de 0,01 y el optimizador Adam. En cuanto a la estructura de la red, se tiene:

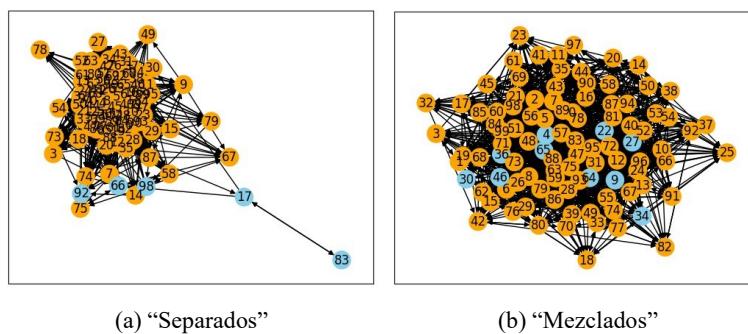
1. Dropout: rate = 0,5.
2. Capa GCN intermedia: 16 neuronas y función de activación ReLU.
3. Dropout: rate = 0,5.
4. Capa GCN final: 2 neuronas y función de activación Softmax.

A la hora de entrenar la red, se optó por un esquema  $k$ -fold, con  $k = 10$ , esto es para cada prueba se entrenaron diez modelos, dejando un grafo diferente para testeo cada

vez. Además, para evaluar el desempeño de la GCN, se observaron ciertas métricas que se describen a continuación.

### 3.2.3 Métricas utilizadas

En un problema de clasificación binaria se puede evaluar el desempeño del modelo a partir de varias métricas. Se toma como referencia la matriz de confusión que puede construirse a partir de la comparación entre las etiquetas reales y las predicciones hechas por la red, y se utilizan las siguientes dos métricas para evaluar el desempeño de la GCN:



**Figura 2.** Ejemplos de grafos con clases no balanceadas. A la izquierda, la clase minoritaria es del 5%, mientras que a la derecha es del 10%.

- AUC (Área bajo la curva ROC): la curva ROC muestra la relación de la tasa de VP contra la tasa de FP.
- Especificidad: proporción de predicciones correctas de clase 1, dado por  $VN/(VN+FP)$ .

Vale aclarar que no se tomaron las métricas exactitud (*accuracy*) y precisión para evaluar las predicciones, ya que no siempre son representativos, especialmente cuando se tienen conjuntos de datos con clases no balanceadas. Por otro lado, como el interés está puesto principalmente en la clase 1, que coincide con la clase minoritaria en los casos no balanceados, se tomó la especificidad (*specificity*) y no su análogo con la clase positiva, conocido como *recall*.

Una vez descriptos los tipos de grafos, la red y las métricas a evaluar, se presentan ahora los principales resultados obtenidos.

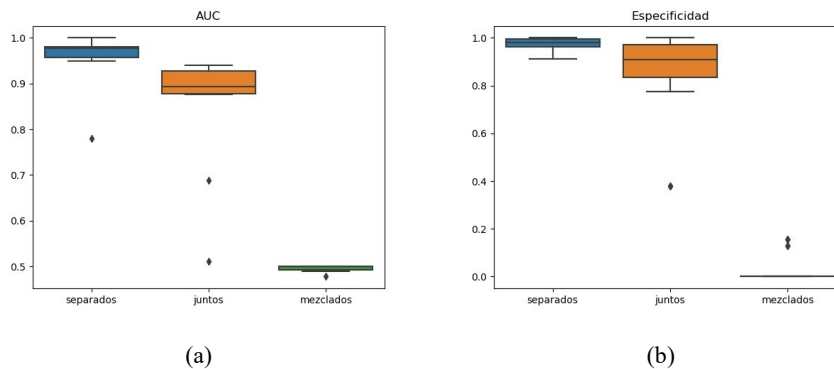
## 4. Resultados y discusión

Se comienza entrenando una red neuronal GCN, usando los mismos hiperparámetros descriptos en la sección anterior, pero variando los conjuntos de entrenamiento, para poner a prueba cada una de las hipótesis formuladas.

### 4.1 Comportamiento según el nivel de separación entre clases

Iniciando por los grafos con clases balanceadas, se pone a prueba la GCN variando el “nivel de separación” entre clases, usando las *features* de los nodos de cada grafo sin modificar. Como de cada experimento se corrieron diez entrenamientos variando en cada caso el conjunto de testeo, se muestran en la Figura 3 las gráficas boxplot, realizadas a partir de las predicciones sobre los diferentes conjuntos de testeo, de los valores AUC (a) y especificidad (b).

Como se esperaba, se observa que la GCN tiene un mejor comportamiento cuando los nodos pertenecen a clases separables, logrando valores de AUC y especificidad mayores a 0,95, mientras que cuando se empiezan a juntar las clases baja el rendimiento de la red a  $\sim 0,8$ , y por el contrario, cuando los nodos están completamente mezclados la GCN no logra identificarlos correctamente, obteniendo entonces un AUC promedio de 0,5 y una especificidad de 0.



**Figura 3.** Valores de AUC y especificidad obtenidos sobre las predicciones realizadas en los diferentes conjuntos de testeo, para diferentes niveles de separación entre clases. Nótese que la escala del eje de las ordenadas de cada grafica es diferente.

### 4.2 Comportamiento al aplanar las características de los nodos

En segundo lugar se puso a prueba el comportamiento de la red al aplanar las características de los nodos a un mismo valor constante, en este caso a 1, pero manteniendo el nivel de separación de las clases y la topología de los grafos utilizados en los experimentos de la sección anterior.

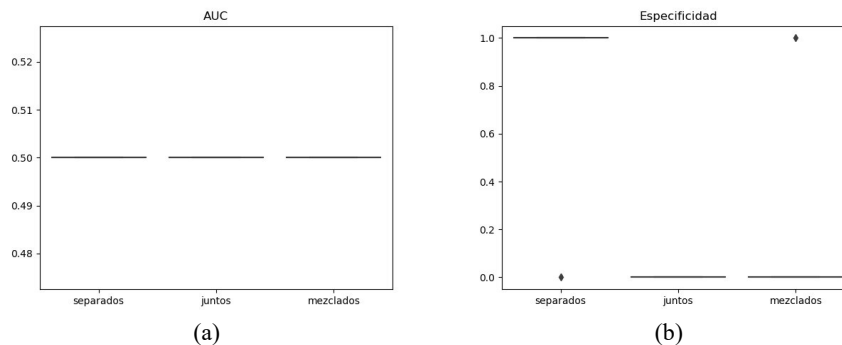
Como se puede observar en la Figura 4, al forzar que todas las características de los nodos de los grafos tengan el mismo valor constante, se pierde la capacidad de la red de diferenciar entre clases. Es decir, a pesar de lo que se esperaba, la topología de los grafos usados durante el entrenamiento no es suficiente para que la GCN logre diferenciar entre nodos. Se observa en el gráfico (a) que el valor de AUC fue 0,5 para cada una de las corridas, mientras que el gráfico (b) se aprecia que la especificidad fue de 0 o 1, lo que indica que la GCN predijo la totalidad de los nodos como pertenecientes a sólo una de las clases.

#### 4.3 Comportamiento ante clases no balanceadas

Por último, cuando se tienen clases que no están balanceadas, la red tiene un rendimiento mucho menor. Se puede observar en la Figura 5 que cuando la clase minoritaria representa un porcentaje menor al 5% del total de nodos, la GCN clasifica a todos los nodos como pertenecientes a una única clase.

Sin embargo, al aumentar el porcentaje de la clase minoritaria la red comienza a aprender a diferenciar entre clases, aunque el desempeño logrado es menor al caso balanceado.

Este comportamiento se observa para los grafos construidos a partir de la configuración “separados”. Por otro lado, de manera análoga a los resultados de las secciones anteriores, para la configuración “mezclados” la red no logra aprender y clasifica a todos los nodos como una única clase. Estas graficas no se muestran por ser similares a las de la Figura 4.



**Figura4.** Valores de AUC y especificidad obtenidos sobre las predicciones realizadas en los diferentes conjuntos de testeo, para diferentes niveles de separación entre clases, al entrenar aplanando las características de los nodos a un mismo valor constante. Nótese que la escala del eje de las ordenadas de cada grafica es diferente.

## 5. Conclusiones y trabajo futuro

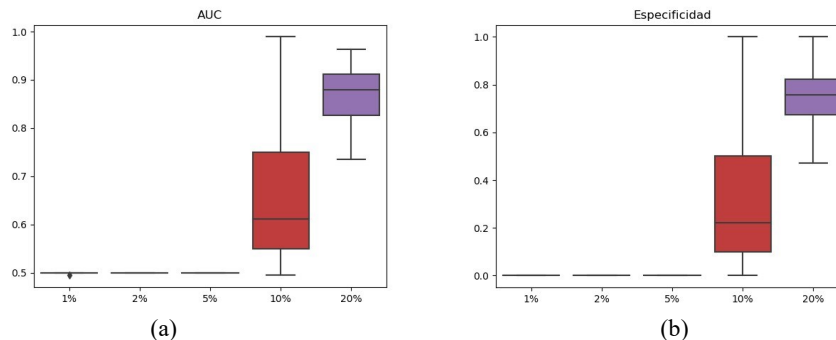
En este trabajo se realizó una exploración de las redes GCN, para analizar cómo funcionan y ver su comportamiento ante diferentes escenarios, dados por distintas configuraciones de grafos.

Como se esperaba, cuando la topología del grafo es tal que las clases son separables entre sí, la GCN logra un muy buen rendimiento, aprovechando no sólo las características de cada nodo, sino también la información aportada por los vecinos. A su vez, ante problemas con una marcada desproporción entre clases, la red no logra distinguir la clase minoritaria. Sin embargo, este comportamiento comienza a mejorar a medida que aumenta el porcentaje de nodos en dicha clase. Por otro lado, a pesar de lo esperado, la topología del grafo no es información suficiente para la GCN.

A partir de lo anterior se concluye que, a la hora de entrenar una red GCN, se verifique en primer lugar que los nodos de los grafos tengan características lo suficientemente “buenas” para obtener mejores resultados.

Como trabajo a futuro se proponen diferentes pruebas que complementarían este trabajo, a fin de alcanzar un mayor entendimiento de este tipo de redes, y que no se llevaron a cabo por limitación en el tiempo. Algunas ideas de pruebas a realizar:

- Ante casos de *features* uniformes, evaluar si el considerar características como el grado (u otras relativas a la estructura del grafo) de los nodos agrega información relevante, y mejora el desempeño de la red.
- Poner a prueba diferentes estrategias de aumento de datos, o función de pérdida ponderada, para evaluar el comportamiento de la red en casos de clases no balanceadas.
- Probar modificaciones en la estructura de la GCN, así como generar grafos con otros tipos de topologías.



**Figura 5.** Valores de AUC y especificidad obtenidos sobre las predicciones realizadas en los diferentes conjuntos de testeo, para grafos con clases no balanceadas. Se probaron diferentes configuraciones, donde la clase minoritaria (la clase de interés) representa el 1,2,5,10 o 20 por ciento del total de nodos. Nótese que la escala del eje de las ordenadas de cada gráfica es diferente.

## Referencias

1. Carson, B.S.: A survey of graph neural networks on synthetic data (2023)
2. Grattarola, D., Alippi, C.: Graph neural networks in tensorflow and keras with spektral [application notes]. IEEE Computational Intelligence Magazine 16(1), 99–106 (2021)
3. Hamilton, W.L.: Graph representation learning. Synthesis Lectures on Artificial Intelligence and Machine Learning 14(3), 1–159 (2020)
4. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)