Aplicación de redes neuronales de grafos: Predicción de enlaces usando el Framework SEAL y modelos GCN y DGCNN

Iván Andrés Espinosa
Departamento de Matemáticas
Universidad Nacional de Colombia
Bogotá D.C., Colombia
iespinosa@unal.edu.co

Edwin Andrés Mora
Departamento de Matemáticas
Universidad Nacional de Colombia
Bogotá D.C., Colombia
edmorag@unal.edu.co

***Resumen*—En este articulo exploraremos el potencial de las redes neuronales de grafos en la resolución del problema de la predicción de enlaces, cuya aplicación es usada en una amplia variedad de campos. Para este fin estudiaremos el framework propuesto por Zhang y Chen, SEAL, que permite emplear modelos como GCN y DGCNN, probarlos, compararlos y medir su rendimiento.**

***Palabras clave*—Grafos, enlaces, etiquetas, mensajes**

I. Introducción

I-A. Breve introducción a grafos

Un grafo modela entidades y las conexiones entre ellas. Podemos definir un grafo $G = (V, E)$ es un conjunto, no vacío V, de nodos conectados a través de aristas en el conjunto $E \subseteq |V| \times |V|$; notamos las aristas de acuerdo a los nodos que une, por ejmplo, si los nodos u y v están relacionados, notaremos a la arista que los une como (u, v). Comúnmente un grafo es disperso, esto quiere decir que solo un pequeño subconjunto de las posibles aristas está presente. Por su parte, un multigrafo, es una generalización del grafo, que permite multiples aristas entre el mismo par de nodos.

Muchos objetos del mundo real pueden ser representados mediante grafos. Por ejemplo, una red social puede verse como un grafo donde los nodos representan personas, y las aristas determinan si hay una relación de amistad entre ellas. Una red de citación académica puede verse tambien como un grafo, considerando los autores de cada artículo como nodos, y la referencia en un artículo a otro autor, la arista que los une. De hecho, este último hace referencia a un multigrafo, ya que un autor puede tener dos articulos distintos donde referencie al mismo autor, formando así, dos aristas distintas que los conectan entre sí.

Un grafo puede ser clasificado como dirigido, en el que las aristas tienen un sentido definido, es decir, para dos nodos u y v que están conectados, no es igual la arista (u, v) a la arista (v, u). O no dirigido, en donde las aristas son relaciones simétricas que no apuntan en ningún sentido, esto es tal que para dos nodos u y v, la arista (u, v) es igual a la arista (v, u). En los dos ejemplos previamente propuestos, el primero hace referencia a un grafo no dirigido, pues la relación de amistad entre dos nodos es simétrica; mientras que el segundo ejemplo es no dirigido, pues la cita de un autor A a otro autor B no implica que tambien exista la cita del autor B al autor A.

Adicionalmente, los nodos o las aristas pueden contener información asociada a ellos mismos. En la red social una persona puede ser caracterizada por sus intereses; mientras que en la red de citación las aristas pueden contener información como el año de escritura del artículo. La información de un nodo está almacenada en un *node embedding*, mientras que la información de una arista está almacenada en una *edge embedding*.

Finalmente, un grafo con N nodos es representado por su matriz de adyacencia $\mathbf{A}$, la cual es una matriz de tamaño $N \times N$ donde la entrada $(a_{m,n})$ es fijada como 1 si existe la arista de m a n, (m, n), o cero de lo contrario. Veamos que si es un grafo no dirigido, esta matriz siempre será simétrica, pues si (m, n) existe, también (n, m).

Así, un grafo G puede ser codificado por tres matrices: $\mathbf{A}$, la matriz de adyacencia que encierra la estructura del grafo, $\mathbf{X}$ la matriz de los *node embeddings* y $\mathbf{E}$ la matriz de las *edge embeddings*.

I-B. Predicción de enlaces

Para un grafo, el problema de predicción de enlaces consiste en, con base a la información presente, pronosticar que aristas entre dos nodos podrían formarse. En el ejemplo de la red social, la predicción de enlaces ayuda a la red a sugerir amistad de una persona a otra, en la red de citación, la predicción de enlaces podría sugerir artículos de interés a una persona con base en los que ya ha citado.

Formlamente definimos este problema como:

Considerando una red $G = (V, E)$ donde V representa los nodos en la red y $E \subseteq |V| \times |V|$ representa el conjunto de enlaces (*aristas*) "verdaderos" entre los nodos de la red. Se nos da el conjunto de nodos V y un subconjunto de los enlaces verdaderos, a los cual nos referimos como enlaces observados. El objetivo de la predicción de enlaces es, entonces, identificar los enlaces verdaderos no observados. En la formulación temporal de la predicción de enlaces, los enlaces observados

corresponden a enlaces verdaderos en un tiempo $\mathbf{t}$, y el objetivo es predecir el conjunto de enlaces verdaderos en un tiempo $\mathbf{t+1}$.

Una primera clase de enfoques, simples pero efectivos para la solución el problema, son los llamados métodos heurísticos, estos utilizan algunas funciones de puntaje, tales como la distancia de grafos, vecinos en común, el indice de Katz, el coeficiente de Jaccard, Adamic/Adar, entre otros, para medir la similitud de los enlaces y determinar donde podría existir uno verdadero. Sin embargo su efectividad por si solos es limitada.

I-C. Estructura del artículo

Para finalizar la introducción, se explicará muy brevemente cómo está estructurado el artículo. Así, después de la introducción, se describe la estructura de una red neuronal de grafos, su arquitectura, y los modelos utilizados en las pruebas. A continuación, se detallan los aspectos de la implementación, incluyendo las adaptaciones de código realizadas. Finalmente, se presentan los resultados obtenidos aplicando los modelos a un dataset de actores, se comparan los resultados y se discuten sus características para, finalmente, formular las conclusiones.

II. Redes neuronales de grafos

Una red neuronal de grafos toma los "node embeddings" $\mathbf{X}$ y la matriz de adyacencia $\mathbf{A}$ y los pasa a través de una serie de *K* capas. Los "node embeddings" son actualizados en cada capa para crear representaciones "ocultas" intermedias $\mathbf{H}_k$ antes de finalmente computar el "output embedding" $\mathbf{H}_K$.

En el comienzo, esta red contiene información de cada nodo por si mismo , para que al final contenga información del nodo y tambiel el contexto en el que está.

Si un grafo contiene tambien "edge embeddings", lo que hace la red en primer lugar es añadir nuevos nodos que representen cada arista y luego conecta los nuevos nodos si las aristas originales compartían nodo. El grafo resultante es denominado grafo de aristas y posteriormente es procesado como si de un grafo común se tratara.

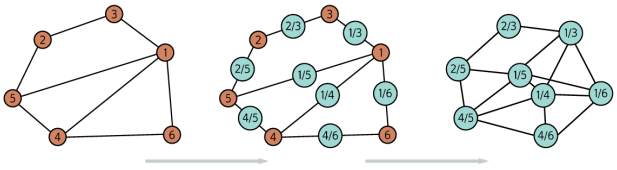

Figura 1: Edge graph, imágen tomada de [1]

(a) Grafo con seis nodos. (b) Se asignan nodos (color cyan) a las aristas originales y (c) se conectan los nuevos nodos si en las aristas originales existía un nodo en común.

Hay muchos tipos de redes neuronales de grafos, sin embargo nos centraremos en las redes neuronales convolucionales de grafos, o GCN para abreviar. Estos modelos son convolucionales en el sentido de que por cada actualización, agregan información de los nodos cercanos, mientras inducen pesos que priorizan la información de los vecinos. Este paso de información descrito, es denominado paso de mensajes. Estos modelos están, además, basados en el espacio, ya que usan la estructura original del grafo.

Cada capa de una GCN es una función $\mathbf{F}[\bullet]$ con parámetros $\mathbf{\Phi}$ que toma los "node embeddings" y la matriz de adyacencia y da como resultado nuevos "node embeddings". Una GCN es entonces escrita como:

$$\mathbf{H}_1 = \mathbf{F}[\mathbf{X}, \mathbf{A}, \phi_\mathbf{0}]$$

$$\mathbf{H}_2 = \mathbf{F}[\mathbf{H_1}, \mathbf{A}, \phi_\mathbf{1}]$$

$$\mathbf{H}_3 = \mathbf{F}[\mathbf{H_2}, \mathbf{A}, \phi_\mathbf{2}]$$

$$\vdots$$

$$\mathbf{H}_K = \mathbf{F}[\mathbf{H_{k-1}}, \mathbf{A}, \phi_\mathbf{K-1}]$$

II-A. Etiquetado de nodos

Definición: Para un grafo $G = (V, E)$, dados dos nodos $x, y \in V$, el subgrafo adjunto de h-saltos para (x, y) es el subgrafo $G^h_{x,y}$ inducido desde G al conjunto de nodos $\{i | d(i, x) \leq h$ o $d(i, y) \leq h\}$.

Así, el subgrafo adjunto describe el "entorno aledaño" de (x, y).

Un etiquetado de nodos es una función $f_l : V \to \mathbb{N}$ donde V es el conjunto de nodos. Así, esta función asigna una etiqueta entera $f_l(i)$ (que se guarda como el primer componente del "node embedding") a todo nodo i en el subgrafo adjunto. El prpósito es usar diferentes etiquetas para marcar nodos con diferentes roles en el subgrafo adjunto: 1) Los nodos centrales x y y son los nodos objetivo entre los cuales el enlace se localiza. 2)Nodos con una posición relativa al centro tienen diferente importancia estructural respecto al enlace. Sin un buen etiquetado de nodos se pierde información estructural.

Así pues, en [2] Zhang y Chen proponen un etiquetamiento de nodos de doble radio o DRNL por sus siglas en inglés. En este, se asignan con la etiqueta 1 a x y a y. Luego, los nodos i con $(d(i, x), d(i, y)) = (1, 1)$, son asignados con la etiqueta $f_l(i) = 2$. Los nodos con radios (1,2) o (2,1) obtienen la etiqueta 3. Los nodos con radios (1,3) o (3,1) obtienen 4. Los nodos con radios (2,2) obtienen 5. Los nodos con radios (1,4) o (4,1) obtienen 6. Los nodos con radios (2,3) o (3,2) obtienen 7. Y así sucesivamente. Lo que se hace básicamente es asignar iterativamente etiquetas más grandes a nodos con radios más grandes con respecto a los dos nodos centrales. Veamos que este etiquetamiento da etiquetas iguales a nodos en la misma "órbita", por lo que el etiquetamiento puede reflejar la posición relativa de los nodos y su importancia estructural en el subgrafo.

II-B. Modelo GCN

Este modelo fue propuesto por Thomas N. Kipf y Max Welling en [3]. Este modelo propone una arquitectura convolucional que opera directamente en los grafos. El modelo fue propuesto originalmente para resolver el problema de clasificación de nodos, sin embargo tambien puede ser usado en la predicción de enlaces. Este propone la siguiente ley de propagación:

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

Aquí $\tilde{A} = a + I_N$ es la matriz de adyacencia del grafo con bucles añadidos. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ y $W^{(l)}$ es una matriz de pesos entrenable específica de la capa. $\sigma(\cdot)$ denota la función de activación, tal como la ReLu. $H^{(l)} \in \mathbb{R}^{N\times D}$ es la matriz de activaciones de la l-ésima capa; $H^{(0)} = X$

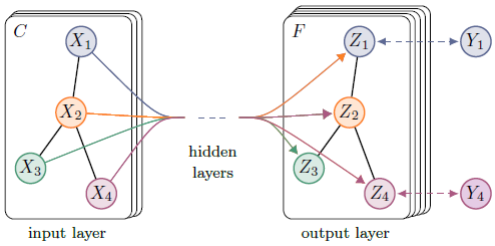

Figura 2: Red convolucional de grafos, imágen tomada de [3], con C canales de entrada y F mapas de características en la capa de salida. La estructura del grafo (las aristas se muestran como líneas negras) se comparte entre las capas, las etiquetas se indican con Y_i.

El modelo de red neuronal propuesto se define de la forma:

$$Z = f(X, A) = \text{ softmax}(\hat{A}\ \text{ReLu}\ (\hat{A}XW^{(0)})W^{(1)})$$

Donde primero debemos calcular $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. Además, $W^{(0)} \in \mathbf{R}^{C\times H}$ es una matriz de pesos de entrada a oculta para una capa oculta con H mapas de caracteristicas. $W^{(1)} \in \mathbf{R}^{H\times F}$ es una matriz de pesos de oculta a salida.

La función de activación softmax se define como:

$$\text{softmax}(x_i) = \frac{1}{Z}exp(x_i) \text{ con } Z = \sum_i exp(x_i)$$

El error es posteriormente evaluado con la función de pérdida de entropía cruzada:

$$(L) = -\sum_{l\in \mathcal{Y}_l}\sum_{f=1} FY_{lf}\ln Z_{lf}$$

Donde $\mathcal{Y}_l$ es el conjunto de indices de nodos que tienen etiquetas.

Los pesos de la red neuronal $W^{(0)}$ y $W^{(1)}$ son entrenados usando el método del gradiente descendente.

II-C. Modelo DGCNN

Este modelo (Redes neuronales convolucionales profundas de grafos, por sus siglas en inglés) fue propuesto por Muhan Zhang, Zhicheng Cui, Marion Neumann y Yixin Chen en [4]. En este articulo proponen un modelo convolucional de grafos localizado y una nueva capa de SortPooling en la cual ordena los nodos del grafo de manera consistente tal que las redes neuronales tradicionales puedan ser entrenadas en grafos.

Este modelo se basa en la observación de que los grafos generalmente carecen de una representación tensorial con un orden fijo, lo cual limita la aplicabilidad de las redes neuronales en grafos. El articulo propone una nueva arquitectura que pueda guardar mucha más información de los nodos y aprender de la topología global del grafo.

Este modelo cuenta con tres etapas secuenciales: 1) La convolución de grafos extrae las caracteristicas subestructurales locales y define un ordenamiento de nodos consistente; 2) Una capa de sortpooling ordena las caracteristicas de los nodos bajo el orden previamente definido y unifica el tamaño de los inputs; 3) Capas convolucionales y densas tradicionales leen la representación ordenada del grafo y hacen predicciones.

Así, los detalles del modelo son definidos como sigue:

Las capas convolucionales son de la forma:

$$Z = f(\tilde{D}^{-1}AXW)$$

Donde $\tilde{A} = A + I$ es la matriz de adyacencia del grafo con auto-bucles, $\tilde{D}$ es la matriz diagonal con $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $W \in \mathbb{R}^{c\times c'}$ es una matriz de parametros convolucionales entrenables para el grafo, f es una función de activación no lineal, y $Z \in \mathbb{R}^{n\times c'}$ es la matriz de activación de salida.

La convolución de grafos agrega información de los nodos en su vecindario local para extraer la información de la subestructura local. Para extraer caracteristicas de la subestructura a multiples escalas, apilamos multiples capas de convolución de grafos como sigue:

$$Z^{t+1} = f(\tilde{\mathbf{D}}^{-1}\tilde{A}Z^tW^t)$$

Donde $\mathbf{Z}^0 = X$, $\mathbf{Z}^t$ es el output de la t-ésima capa convolucional, c_t es el número de canales de salida para la capa t, y $\mathbf{W}^t$ mapea los c_t canales a c_{t+1} canales. Luego de multiples capas de convolución , añadimos una capa que concatene la salida $\mathbf{Z}^t$, $t = 1, ..., h$ horizontalmente para formar una salida concatenada escrita como $\mathbf{Z}^{1:h} := [\mathbf{Z}^1, \ ... \ \mathbf{Z}^h]$, donde h es el número de capas convolucionales de grafos. En la salida concatenada $\mathbf{Z}^{1:h}$ cada fila puede considerarse como el descriptor de caracteristicas de un vértice, el cual codifica sus informaciones sobre la subestructura local a multiples escalas.

Luego de tener la salida concatenada aplicamos sobre esta, una capa de sortpooling, cuya función principal es la de ordenar los descriptores de características, cada uno de los cuales representa un nodo, en un orden consistente antes de usarlos para alimentar capas densas y convolucionales de una dimensión.

Posteriormente se aplican varias capas de MaxPooling y convoluciones de una dimensión, para aprender patrones locales en la secuencia de nodos. Finalmente, se añade una capa completamente conectad, seguida de una capa de softmax.

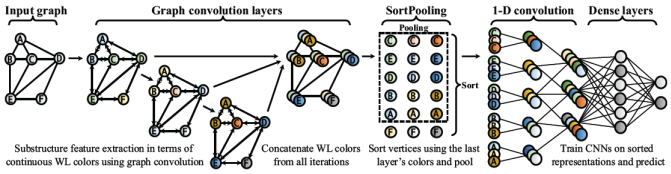

Figura 3: Estructura de la DGCNN, imágen tomada de [4].

II-D. Framework SEAL

El framework SEAL, propuesto por Muhan Zhang y Yixin Chen en [2] aprende las caracteristicas estructurales generales del grafo para la predicción de enlaces. Contiene tres pasos: 1)Extracción de grafos adjuntos, 2) Construcción de la matriz de información de los nodos y 3)Aprendizaje con ayuda de una red neuronal de grafos (GNN). Así, dada una red, el objetivo es aprender automáticamente una "heurística" que mejor explique la formación de enlaces. Esta función toma los subgrafos adjuntos alrededor de un enlace como entradas, y da como resultado que tan probable es que el enlace exista. Para aprender una función tal, se entrena una red neuronal de grafos sobre los subgrafos adjuntos. Así, el primer paso en SEAL es extraer los subgrafos adjuntos para un conjunto muestreado de enlaces positivos (observados) y un conjunto muestreado de enlaces negativos (no observados) para construir los datos de entrenamiento.

El segundo paso es construir la matriz de información de los nodos X que, en SEAL, tiene tres componentes: etiquetas estructurales para los nodos (etiquetado de nodos), "node embeddings" y los atributos de los nodos; cada columna de esta matriz corresponde al vector de caracteristicas de un nodo.

Así, el nombre de SEAL refleja lo que hace el algoritmo. Por sus siglas en inglés SEAL(aprendizaje de subgrafos, embeddings y atributos para predicción de enlaces), enfatizando en su habilidad de aprender de tres distintos tipos de caracteristicas.

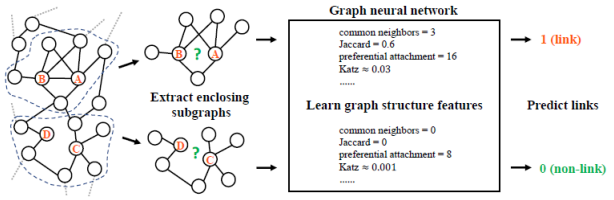

Figura 4: Framework SEAL, imágen tomada de [2]. Para cada enlace objetivo, SEAL extrae un subgrafo local adjunto y utiliza una red neuronal global para aprender las características generales de la estructura del grafo para la predicción de enlaces. Notemos que las heurísticas que se enumeran dentro del cuadro son solo para fines ilustrativos.

III. Implementación

El código original de SEAL se encuentra disponible en el github del autor, allí se encuentra implementado tanto para python como para matlab. Sin embargo, nosotros buscamos implementarlo en google colab. Para esto fue necesario realizar ciertas modificaciones para que corriera adecuadamente.

SEAl funciona principalmente con ayuda de PyTorch y PyTorch Geometric, las cuales, ya implementan por si mismas el modelo GCN; por su parte, el modelo DGCNN se implementa usando como base el modelo GCN para luego usar capas de una dimensión.

El dataset usado lleva por nombre Actor y viene incluído entre los datasets de la librería PyTorch Geometric, esta cuenta con un grafo con 7600 nodos y 30019 aristas, cada nodo se ve descrito por 932 características y puede pertenecer a 5 clases distintas.

Para cada modelo, se realizó un entrenamiento de 50 épocas, la partición para el entrenamiento era del $50\,\%$ del dataset original, la partición para validar contenía un $20\,\%$ del dataset original, y el $30\,\%$ restante se usó para testear el modelo obtenido con el fin de ver que modelo obtenía mejores resultados en la tarea de predecir enlaces.

Al final, el puntaje fue dado usando la métrica AUC.

IV. Resultados

La siguiente tabla contiene los resultados al probar la predicción de enlaces en el dataset Actor con los modelos GCN y DGCNN:

Modelo	Validación	Test
GCN	83.22 %	82.42 %
DGCNN	83.41 %	82.61 %

Cuadro I: Comparación resultados, la columna de validación hace referencia a el puntaje más alto de validación a través de las 50 épocas. La columna test muestra el puntaje obtenido en la última época

Como se puede observar, los resultados indican un desempeño ligeramente mejor por parte del modelo DGCNN.

V. Conclusión

Las redes neuronales de grafos, utilizando el framework SEAL y los modelos GCN y DGCNN, son herramientas efectivas para la predicción de enlaces en grafos, mostrando que ambos modelos tienen un rendimiento similar, aunque DGCNN presenta una ligera ventaja debido a su capacidad para ordenar y unificar representaciones de grafos. La implementación práctica en Google Colab demuestra que estas técnicas pueden aplicarse con éxito en contextos reales, ofreciendo soluciones viables para problemas complejos de predicción en redes.

Referencias

[1] S. J. Prince, *Understanding Deep Learning*. The MIT Press, 2023. [Online]. Available: http://udlbook.com

[2] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 5165–5175.

[3] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *5th International Conference on Learning Representations*, 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1609.02907

[4] M. N. Muhan Zhang, Zhicheng Cui and Y. Chen, "An end-to-end deep learning architecture for graph classification," *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018. [Online]. Available: https://dl.acm.org/doi/abs/10.5555/3504035.3504579

[5] Y. X. K. W. Muhan Zhang, Pan Li and L. Jin, "Labeling trick: A theory of using graph neural networks for multi-node representation learning," *35th Conference on Neural Information Processing Systems (NeurIPS 2021).*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2010.16103

[6] Link prediction algorithms. [Online]. Available: http://be.amazd.com/link-prediction/