GRNNs

Introducción a las redes neuronales de Grafos

Seminario máster IA³

12 junio 2024

IDAL Intelligent Data Analysis Laboratory

Arturo Sirvent Fresneda

Índice seminarios

Día 11 de junio *(1.5 horas)*

- ✓ Teoría de grafos y métricas
- ✓ Ejemplo/práctica para trabajar con grafos en python.
- ✓ Tipos y clasificación de redes.
- ✓ Fundamentos matemáticos.
- ✓ Graph Convolutional Networks (GCNs).
- Práctica sobre GCNs

Día 12 de junio *(1.5 horas)*

- Lo que falte de hoy
- Graph Attention Networks (GANs)
- Práctica sobre GANs
- Técnicas avanzadas
- *GraphSAGE*
- Más aplicaciones prácticas

IDAL Intelligent Data Analysis Laboratory

Índice seminarios

Día 11 de junio *(1.5 horas)*

- ✓ Teoría de grafos y métricas
- ✓ Ejemplo/práctica para trabajar con grafos en python.
- ✓ Tipos y clasificación de redes.
- ✓ Fundamentos matemáticos.
- ✓ Graph Convolutional Networks (GCNs).

Día 12 de junio *(1.5 horas)*

- Práctica sobre GCNs
- Graph Attention Networks (GANs)
- Práctica sobre GANs
- Técnicas avanzadas
- *GraphSAGE*
- Más aplicaciones prácticas

IDAL Intelligent Data Analysis Laboratory

Breve resumen

Lo más importante que vimos ayer

IDAL
Intelligent
Data
Analysis
Laboratory

Redes neuronales de Grafos

3 GNNs (redes neuronales de grafos)

- Fundamentos
- Taxonomía (partes/clasificación)
- Entrenamiento

Hay muchas variantes y a veces cuesta tener una visión nítida de todo, para ello nos centraremos en 4 de los conceptos/arquitecturas centrales:

- **Message Passing Neural Networks (MPNNs)**
  - **Graph Convolutional Networks (GCNs)**
  - **Graph Attention Networks (GATs)**
  - **GraphSAGE**

IDAL Intelligent Data Analysis Laboratory

Nota: Todos estos modelos son formas de aplicar aprendizaje profundo a datos estructurados en forma de grafos, pero cada uno tiene características únicas y aplicaciones específicas.

Redes neuronales de Grafos

3 GNNs (redes neuronales de grafos)

- Fundamentos
- Taxonomía (partes/clasificación)
- Entrenamiento

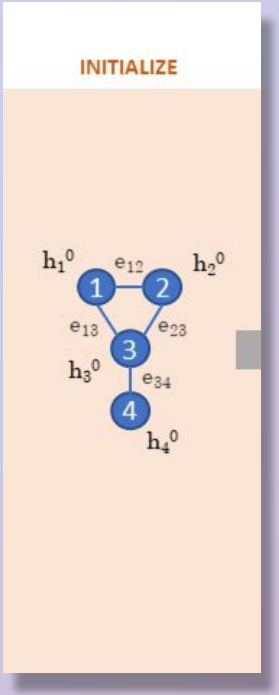
Message Passing Neural Networks (MPNNs)

Las **MPNNs** representan un marco general para muchas de las técnicas de redes neuronales de grafos, incluidas las GCNs, GATs, GraphSAGE, etc.

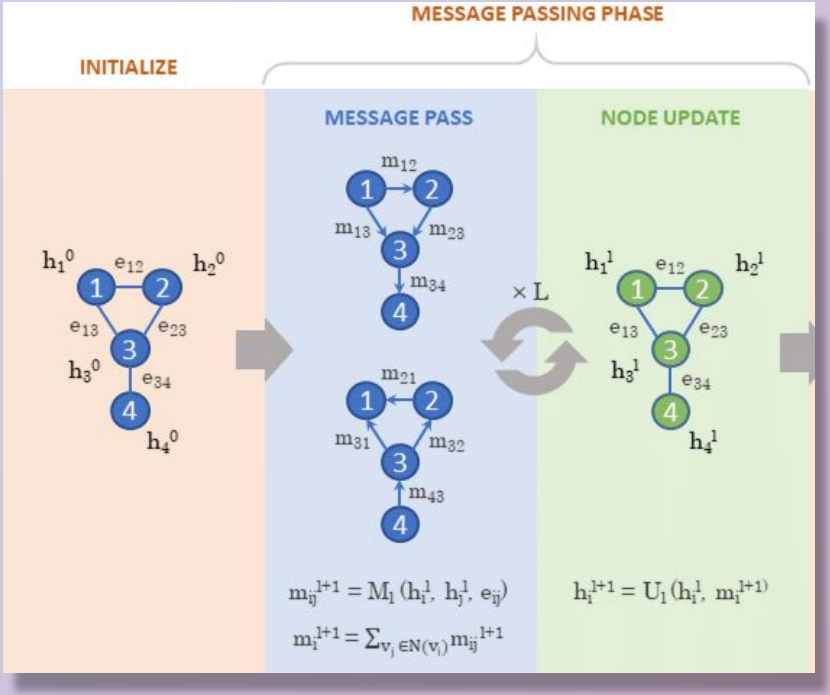
El concepto central es que los **nodos intercambian mensajes con sus vecinos**, y **estos mensajes son luego agregados** y **usados para actualizar los estados de los nodos**.

En este marco, se pueden definir explícitamente las funciones de **generación de mensajes**, **agregación** y **actualización**, lo que ofrece una gran flexibilidad para diseñar diferentes tipos de agregaciones y transformaciones.

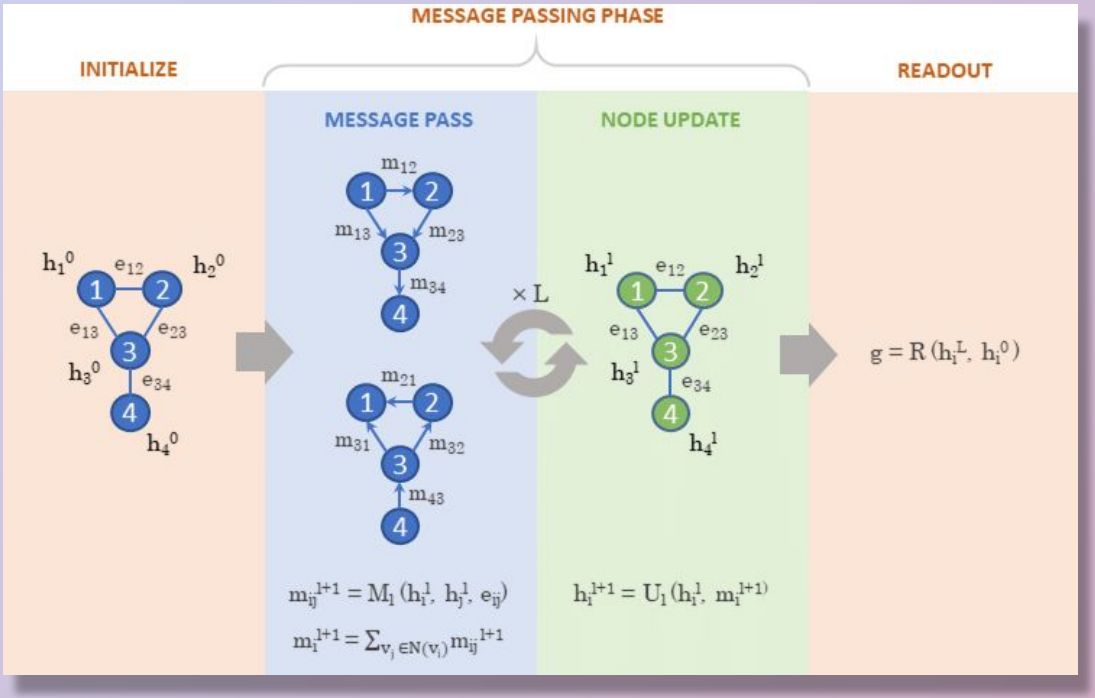
Redes neuronales de Grafos



Redes neuronales de Grafos



Redes neuronales de Grafos



Redes neuronales de Grafos

3 GNNs (redes neuronales de grafos)

- Fundamentos
- Taxonomía (partes/clasificación)
- Entrenamiento

Graph Convolutional Networks (GCNs):

Las **GCNs** son una técnica que generaliza las operaciones de convolución de las CNNs para trabajar sobre grafos.

La idea es utilizar la estructura del grafo para definir una operación de "convolución" que, en lugar de operar sobre una región local de una imagen (como en las **CNNs**), opera sobre los "vecindarios" de un nodo en un grafo.

Esto implica sumar y transformar las características de los nodos vecinos usando pesos compartidos, lo que permite capturar la estructura local del grafo.

Redes neuronales de Grafos

3 GNNs (redes neuronales de grafos)

- Fundamentos
- Taxonomía (partes/clasificación)
- Entrenamiento

Graph Attention Networks (GATs):

Las **GATs** introducen un mecanismo de atención en el proceso de agregación de las características de los vecinos.

En una **GAT**, la influencia de cada vecino en el nodo central se pondera mediante **coeficientes de atención**, que son aprendidos por la red. Esto permite que el modelo dé más importancia a ciertos nodos vecinos en función de cómo de relevantes son para la tarea específica, proporcionando una forma adaptativa y potente de agregación de características.

Redes neuronales de Grafos

3 GNNs (redes neuronales de grafos)

- Fundamentos
- Taxonomía (partes/clasificación)
- Entrenamiento

GraphSAGE:

Es un modelo que también se centra en cómo los nodos agregan características de sus vecinos, pero con un enfoque específico en la eficiencia y la escalabilidad.

En lugar de usar toda la información de los vecinos, **GraphSAGE** emplea un método de muestreo para seleccionar un subconjunto fijo de vecinos y luego utiliza diferentes funciones de agregación (como el promedio, la suma, o el máximo) para combinar sus características.

Esto permite que **GraphSAGE** maneje grafos grandes de manera eficiente.

Redes neuronales de Grafos

3 GNNs (redes neuronales de grafos)

- Fundamentos
- Taxonomía (partes/clasificación)
- Entrenamiento

Relación entre los Conceptos:

Todos estos modelos comparten el principio básico de que la representación de un nodo se puede mejorar mediante la incorporación de información de sus vecinos.

Las **GCNs** y **GATs** son casos específicos dentro del marco general de las **MPNNs** donde las operaciones de agregación y actualización están más estructuradas.

GraphSAGE por otro lado, amplía estos modelos al introducir métodos de muestreo que hacen que el procesamiento de grafos grandes sea más factible.

Cada modelo ofrece diferentes enfoques y mejoras sobre cómo se puede integrar la información de los vecinos en la representación de un nodo, y el uso de uno u otro dependerá de las necesidades específicas de rendimiento, precisión y escalabilidad del problema en cuestión.

Caso concreto

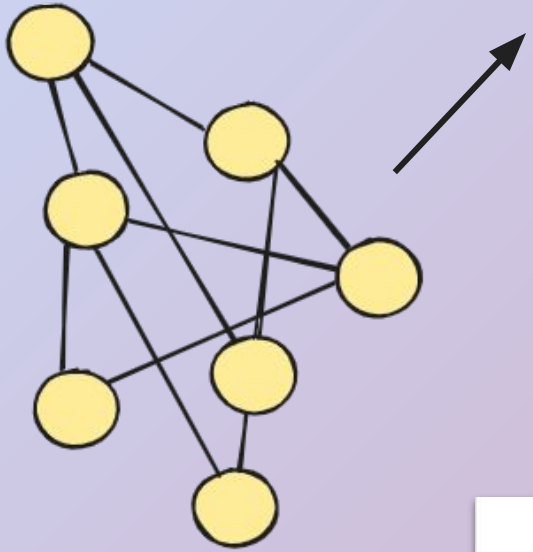
Redes convolucionales de Grafos (GCNs)

¿Cómo trasladar el concepto de convolución a un grafo?

IDAL
Intelligent
Data
Analysis
Laboratory

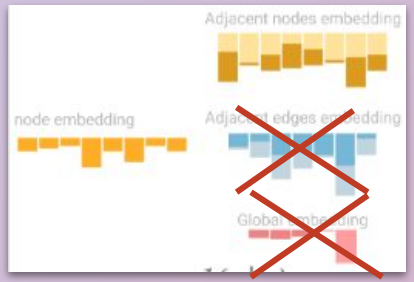
Fundamento matemático en GCNs

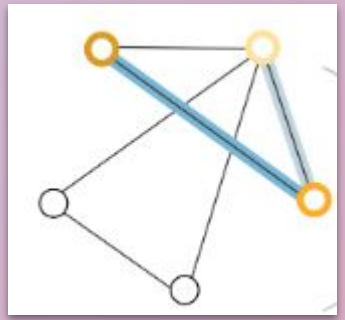
GCN simplificada (“vanilla”)



A: Matriz de adyacencia

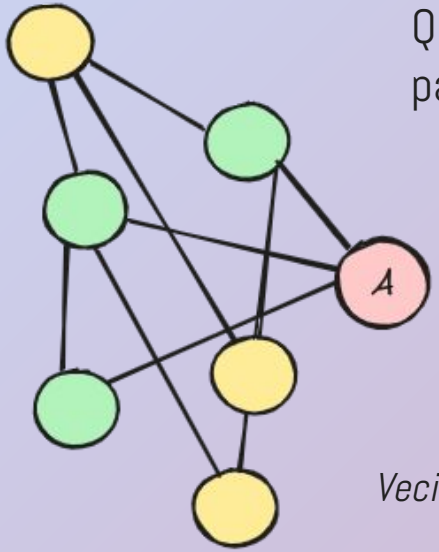
- Nos centramos solo en los nodos
- Su conectividad no es ponderada
- Cada nodo tiene un vector de características asociado.
- No tenemos variables globales





Fundamento matemático en GCNs

GCN simplificada ("vanilla")



Queremos calcular la nueva representación de **A**, usando para ello los vectores de características de sus vecinos:

$$h_A = \sum_{i \in \mathcal{N}_A} x_i W^T$$

Matriz que transforma linealmente el vector de características

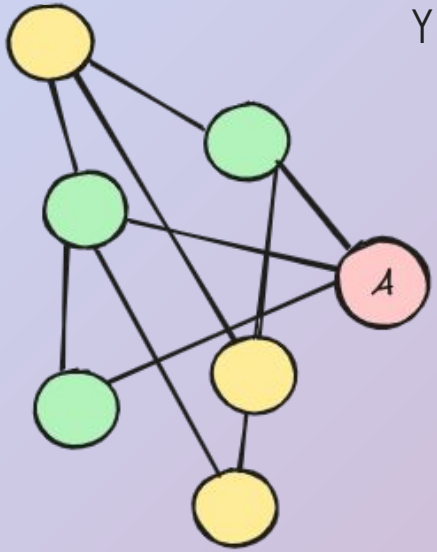
Vecinos (nodos verdes)

Cada vector de características de los vecinos:

ID	Name	Age	Gender
1	Mary	76	Female
2	John	75	Male

Fundamento matemático en GCNs

GCN simplificada ("vanilla")



Y esto lo tendremos que realizar para cada nodo del grafo:

For **A** in **nodos:**

$$h_A = \sum_{i \in \mathcal{N}_A} x_i W^T$$

Mejor y más óptimo con **álgebra matricial**.

$$\tilde{A} = A + I$$

Añadimos la identidad para tener en cuenta el vector de variables del propio nodo a calcular.

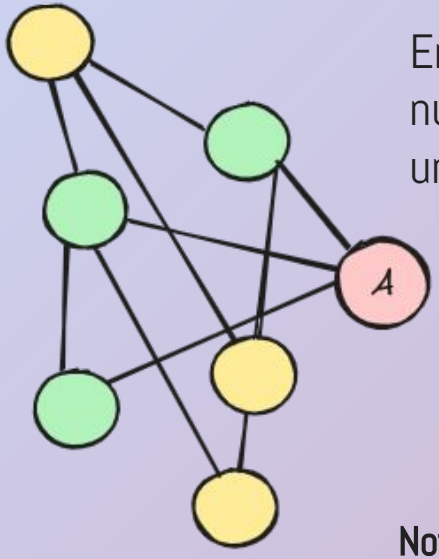
$$H = \tilde{A}^T X W^T$$

Aquí estamos calculando todos los estados intermedios H=(hA,hB,...) de golpe.

IDAL Intelligent Data Analysis Laboratory

Fundamento matemático en GCNs

GCN estandar



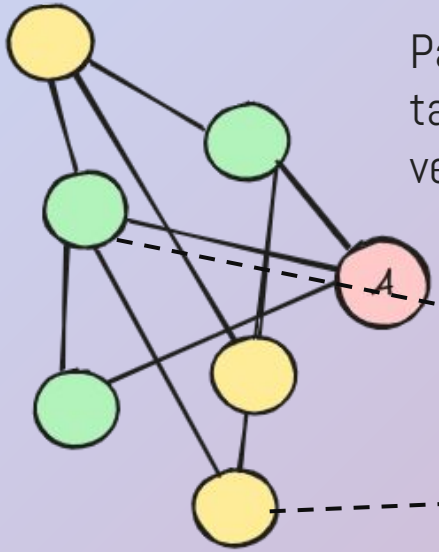
En el caso anterior, no estábamos teniendo en cuenta el número de vecinos que tiene cada nodo. Y al tratarse de una suma sobre vecinos, sería conveniente normalizarlo:

$$h_A = \sum_{i \in \mathcal{N}_A} x_i W^T \quad \rightarrow \quad h_A = \frac{1}{\deg(A)} \sum_{i \in \mathcal{N}_A} x_i W^T$$

Nota: Si un nodo A tiene 1000 vecinos y otro B solo 5, por lo general, el hA será mucho mayor que hB porque la suma se ha hecho 1000 veces, y en B solo 5.

Fundamento matemático en GCNs

GCN estandar

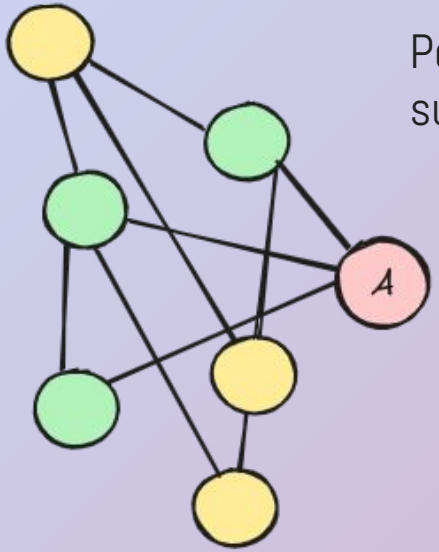


Para hacerlo todo de manera matricial, ahora usaremos también la matriz **D**, que recoge el grado (número de vecinos) de cada nodo.

$$D = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

Fundamento matemático en GCNs

GCN estandar



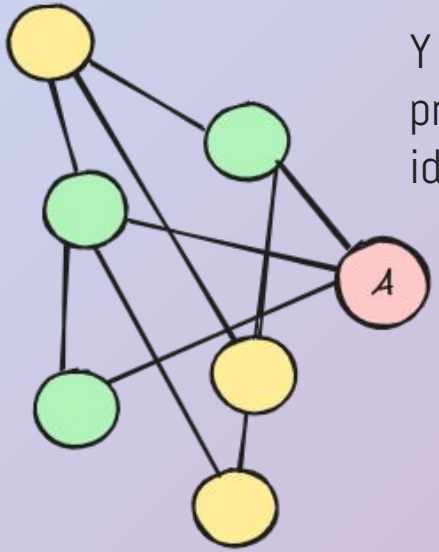
Pero para el cálculo nos interesa 1/deg(i) no el grado, sino su inversa:

$$D^{-1} = \begin{pmatrix} \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} \end{pmatrix}$$

IDAL Intelligent Data Analysis Laboratory

Fundamento matemático en GCNs

GCN estandar



Y al igual que con la matriz de adyacencia, contamos al propio nodo como vecino, por eso debemos de sumarle la identidad, cada nodo tiene un vecino más (él mismo):

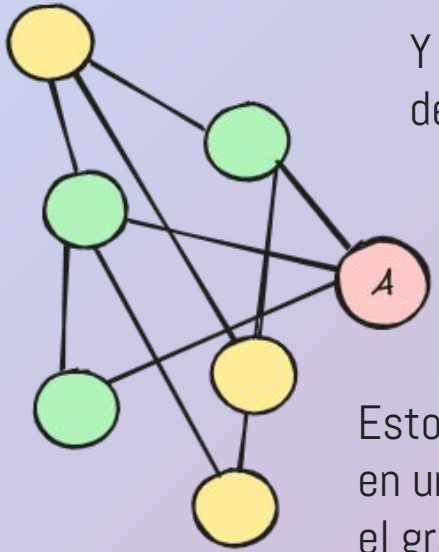
$$\tilde{D}^{-1} = (D + I)^{-1}$$

$$\tilde{D}^{-1} = \begin{pmatrix} \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{5} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} \end{pmatrix}$$

IDAL Intelligent Data Analysis Laboratory

Fundamento matemático en GCNs

GCN estandar



Y aquí tenemos dos maneras opuestas de incluir la matriz de grado en las ecuaciones:

$$\tilde{D}^{-1}\tilde{A}XW^{T}$$

$$\tilde{A}\tilde{D}^{-1}XW^{T}$$

Esto es lo que proponíamos en un principio dividiendo por el grado de cada nodo.

Pero en el artículo original los autores descubrieron que las características de nodos con muchos vecinos, se ven demasiado "diluidas" debido a dividirse por números altos.
Propusieron una alternativa híbrida.

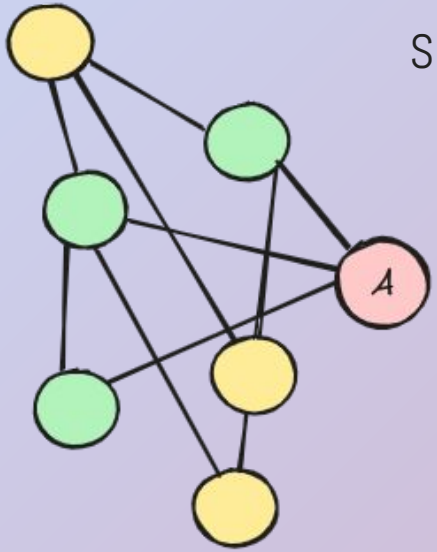
$$H = \tilde{D}^{-\frac{1}{2}}\tilde{A}^{T}\tilde{D}^{-\frac{1}{2}}XW^{T}$$

https://arxiv.org/pdf/1609.02907

IDAL Intelligent Data Analysis Laboratory

Fundamento matemático en GCNs

GCN estandar



Sin expresarlo con álgebra matricial:

$$H = \tilde{D}^{-\frac{1}{2}}\tilde{A}^{T}\tilde{D}^{-\frac{1}{2}}XW^{T}$$

For **A** in **nodos:**

$$h_A = \sum_{i \in \mathcal{N}_A} \frac{1}{\sqrt{\deg(i)}\sqrt{\deg(A)}} x_i W^T$$

Nota: Aquí "A" hace referencia al nodo A, no a la matriz de adyacencia.

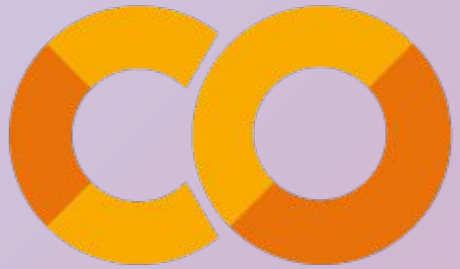
Prácticas

Veamos su aplicación en còdigo

Práctica sobre GCNs

Colab Notebook:

https://drive.google.com/file/d/1d38sO2ZU3OpSMYWT6wE6xHTW6TpSI2J2/view?usp=sharing



Caso concreto

Redes de Grafos con Atención (GANs)

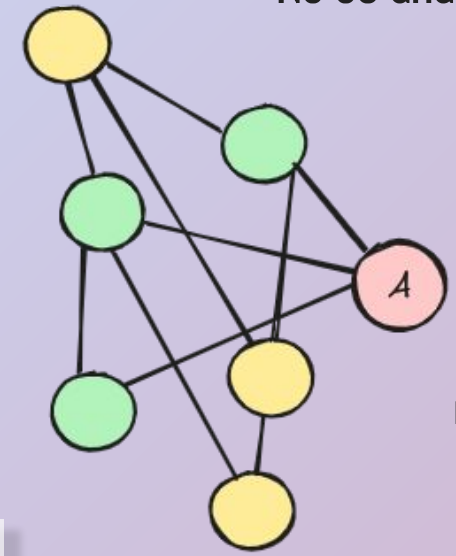
¿Cómo implementar los mecanismos de atención en las GNNs?

IDAL
Intelligent
Data
Analysis
Laboratory

Fundamento matemático en GANs

En el caso anterior, se le prestó la misma atención a cada uno de los vecinos del nodo a calcular:

No es una suma ponderada, cada vecino tiene la misma importancia.



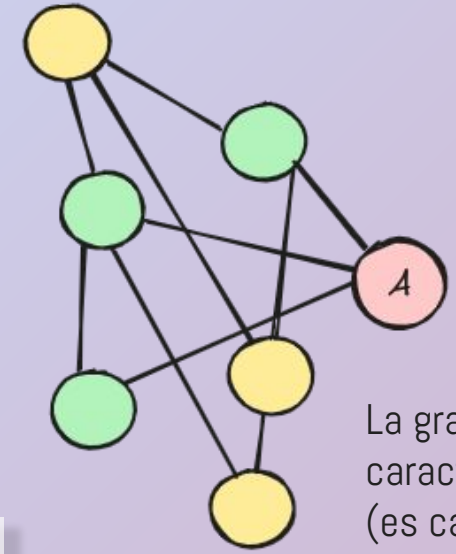
$$h_A = \sum_{i \in \mathcal{N}_A} x_i W^T$$

Pero esto no es cierto en general. A veces unos vecinos serán más importantes que otros. De hecho, se da mayor importancia a los nodos con menos vecinos gracias a la normalización usada:

$$\frac{1}{\sqrt{\deg(i)}\sqrt{\deg(A)}}$$

Fundamento matemático en GANs

El "*approach*" anterior es un problema porque **solo da esa importancia basado en el grado de los nodos**, y **NO en el vector de características de ellos** (parte fundamental en los mecanismos de atención).
Para solucionar esto, añadimos un **término** de hace referencia a la importancia entre el nodo A y el nodo vecino "i".



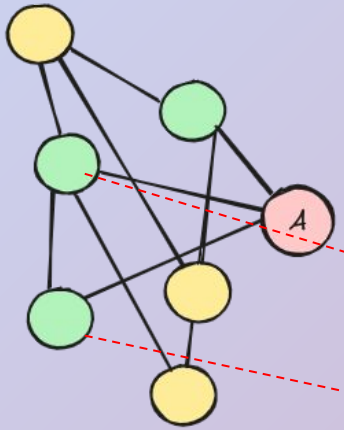
$$h_A = \sum_{i \in \mathcal{N}_A} \alpha_{iA} x_i W^T$$

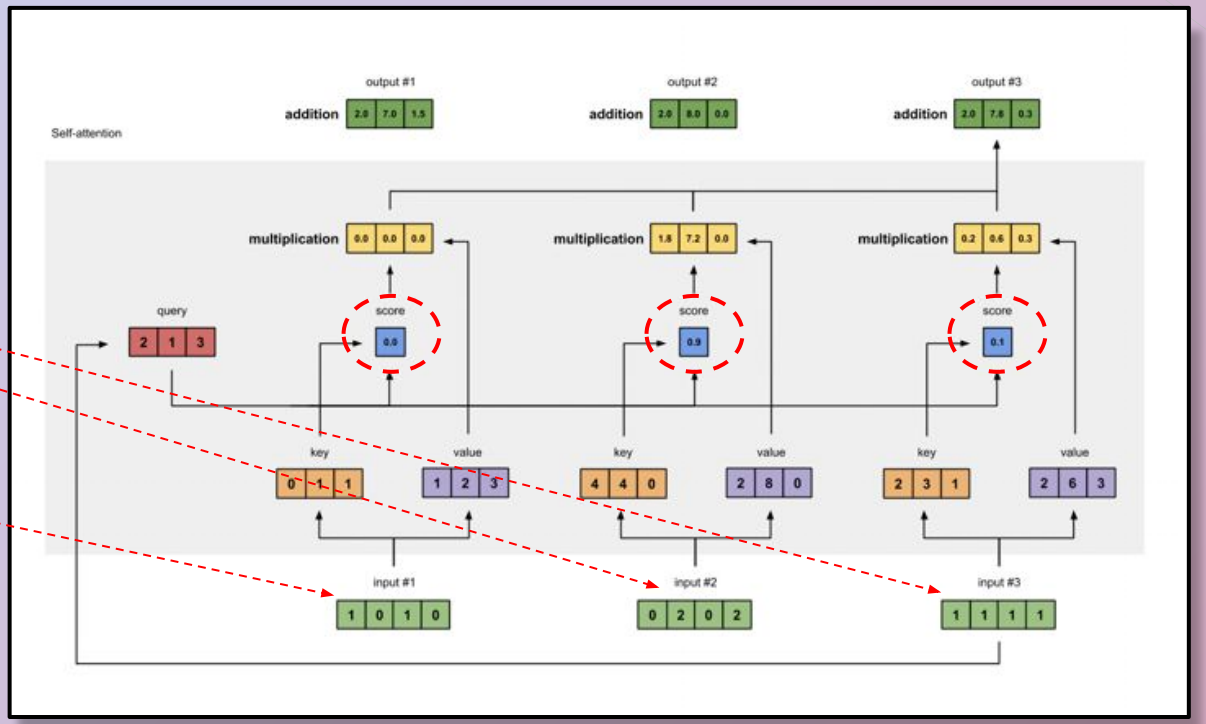
Esto es lo que llamamos: ***"Attention scores"***

La gracia de esto, es que se calculan usando los valores input (vector de características) de cada uno de los nodos. Por eso se llama ***self-attention*** (es calculado entre los propios inputs).

Fundamento matemático en GANs

Breve inciso sobre el mecanismo de ***self-attention***:

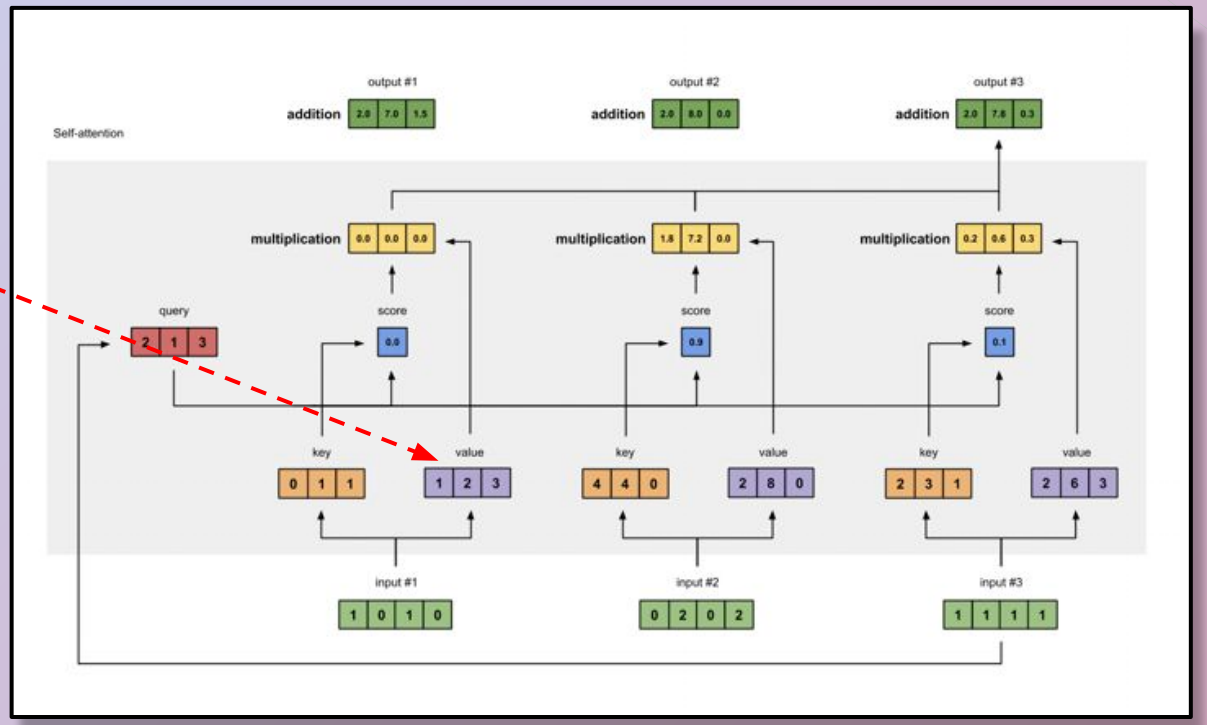




Fundamento matemático en GANs

Breve inciso sobre el mecanismo de ***self-attention***:

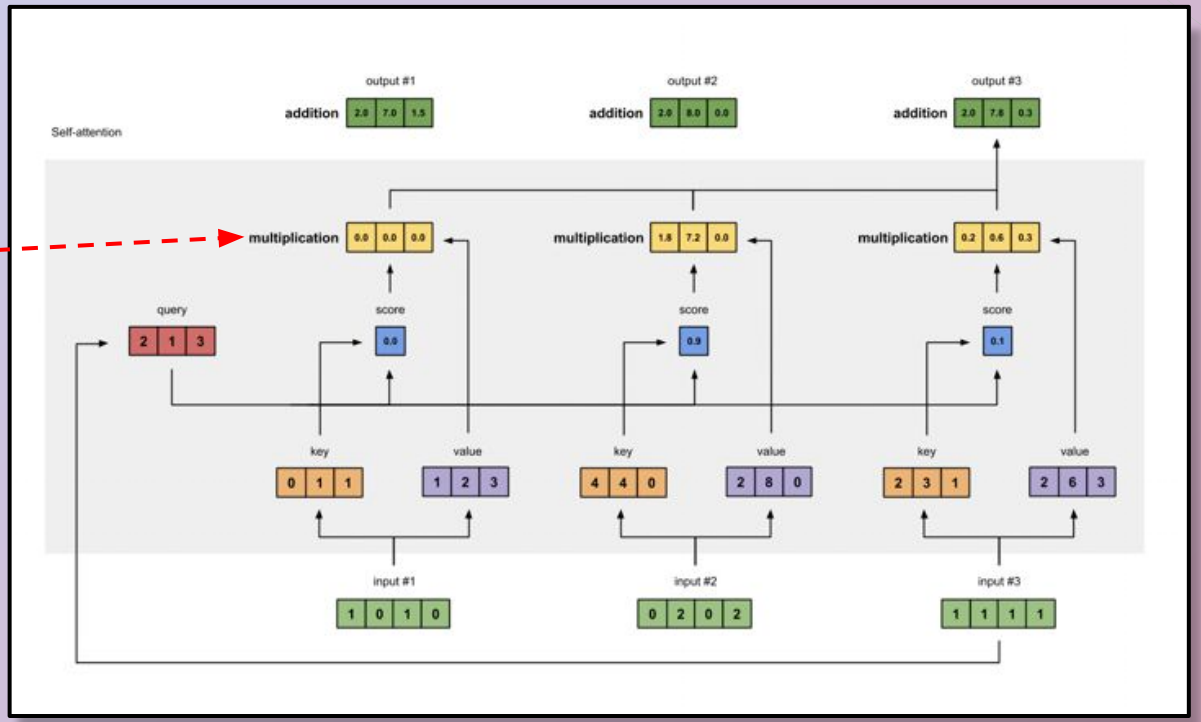
$$h_A = \sum_{i \in \mathcal{N}_A} \alpha_{iA} x_i W^T$$



Fundamento matemático en GANs

Breve inciso sobre el mecanismo de ***self-attention***:

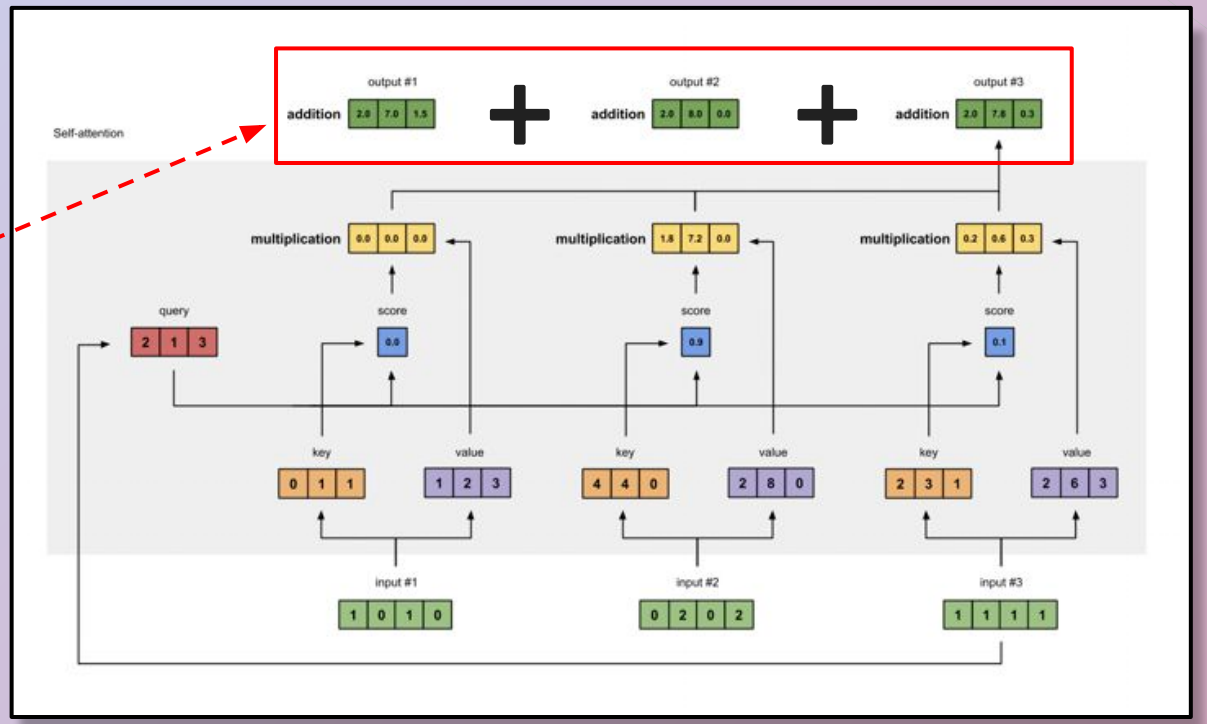
$$h_A = \sum_{i \in \mathcal{N}_A} \alpha_{iA} x_i W^T$$



Fundamento matemático en GANs

Breve inciso sobre el mecanismo de ***self-attention***:

$$h_A = \sum_{i \in \mathcal{N}_A} \alpha_{iA} x_i W^T$$

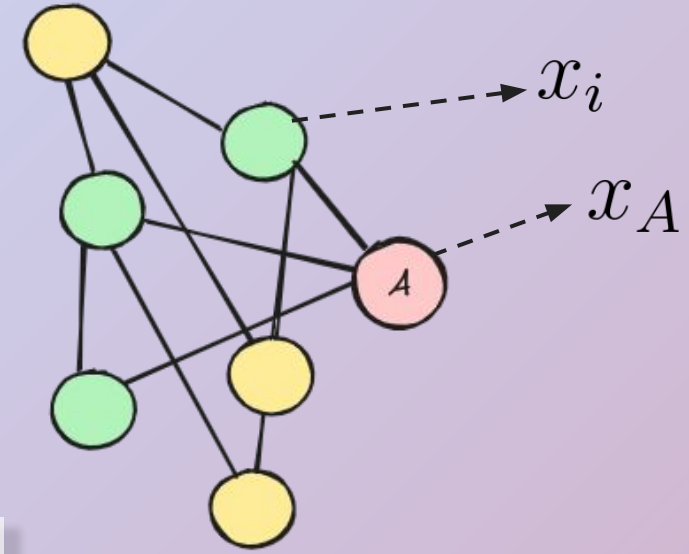


IDAL Intelligent Data Analysis Laboratory

Fundamento matemático en GANs

En nuestro caso no es tanto el uso explícito de: *Query + Key + Value*

Sino que las ***attention scores*** se calcularán de la siguiente manera:



Cada nodo tiene un vector de características

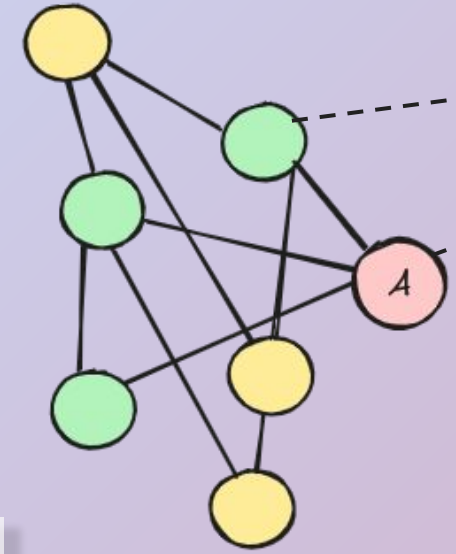
(2.4 , 1 , 0.43 , 32 , 2.1 , ...)

IDAL Intelligent Data Analysis Laboratory

Fundamento matemático en GANs

En nuestro caso no es tanto el uso explícito de: *Query + Key + Value*

Sino que las ***attention scores*** se calcularán de la siguiente manera:



$$x_i \dashrightarrow \boldsymbol{W} x_i \dashrightarrow h_i \dashrightarrow h_A || h_i$$

$$x_A \dashrightarrow \boldsymbol{W} x_A \dashrightarrow h_A \dashrightarrow h_A || h_i$$

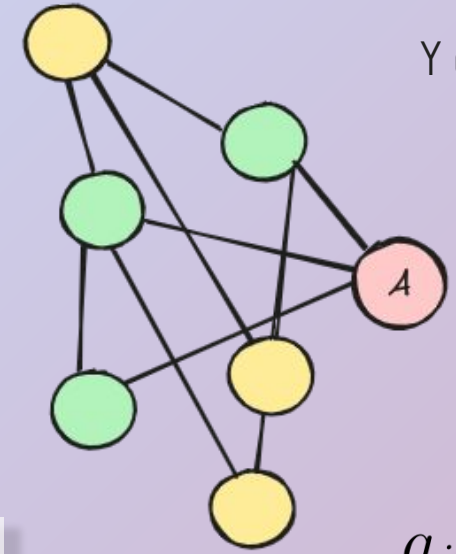
Al igual que en una GCN, se le aplica una transformación lineal (**W**) a cada vector x, creando así su "estado intermedio" o estado oculto.

Una vez tenemos estos vectores intermedios, los concatenamos.

Fundamento matemático en GANs

En nuestro caso no es tanto el uso explícito de: *Query + Key + Value*

Sino que las ***attention scores*** se calcularán de la siguiente manera:



Y el vector concatenado hará las veces de *"Key"*

$$h_A||h_i$$

Y la *"query"*, el vector que multiplicado por la *"key"* decide la importancia que dar dicha conexión nodo-nodo, será aprendida durante el entrenamiento: W_{att}

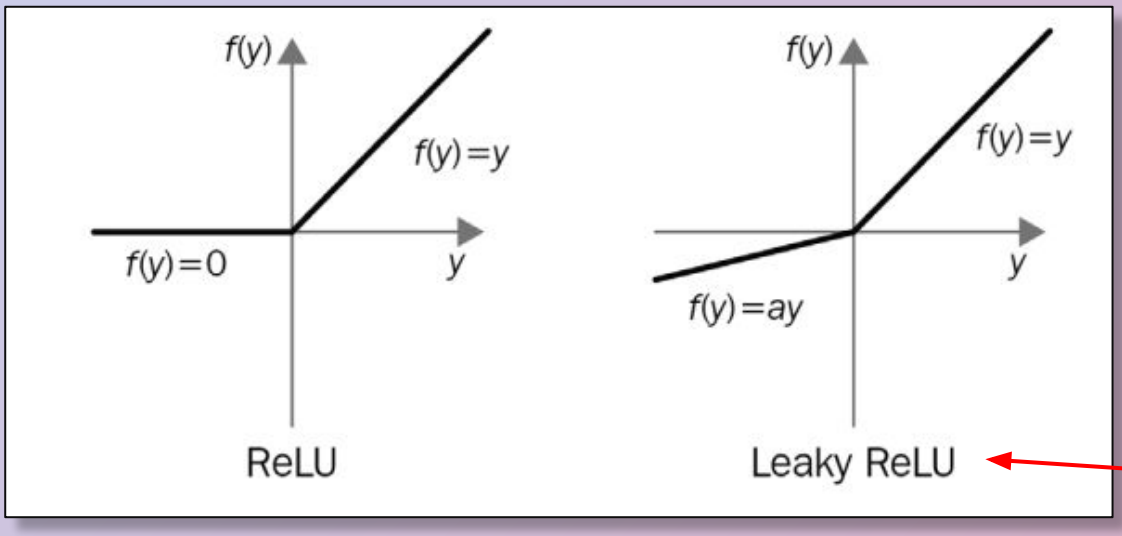
$$a_{iA} = W_{att}^T[h_A||h_i] = \underset{\text{"Query"}}{W_{att}^T}[\underset{\text{"Key"}}{\boldsymbol{W}x_A||\boldsymbol{W}x_i}]$$

Es a no alpha!

IDAL Intelligent Data Analysis Laboratory

Fundamento matemático en GANs

Hasta ahora todas las operaciones que hemos realizado son lineales, pero la no-linealidad es un componente esencial en la redes neuronales. Por eso hay que incluir un función de activación:



Opción usada por los autores

$$e_{iA} = \mathrm{LeakyReLu}(a_{iA})$$

IDAL Intelligent Data Analysis Laboratory

Fundamento matemático en GANs

Pero no está normalizado el peso que le damos a cada conexión, sería conveniente tener acotado y normalizados estos valores.

$$\sum_{i \in \mathcal{N}_A} \alpha_{iA} = 1$$

Para hacer esto, vamos usar una función softmax, muy usada para normalizar probabilidades:

$$\alpha_{iA} = \text{softmax}_A(e_{iA}) = \frac{\exp(e_{iA})}{\sum_{j \in \mathcal{N}_A} \exp(e_{jA})}$$

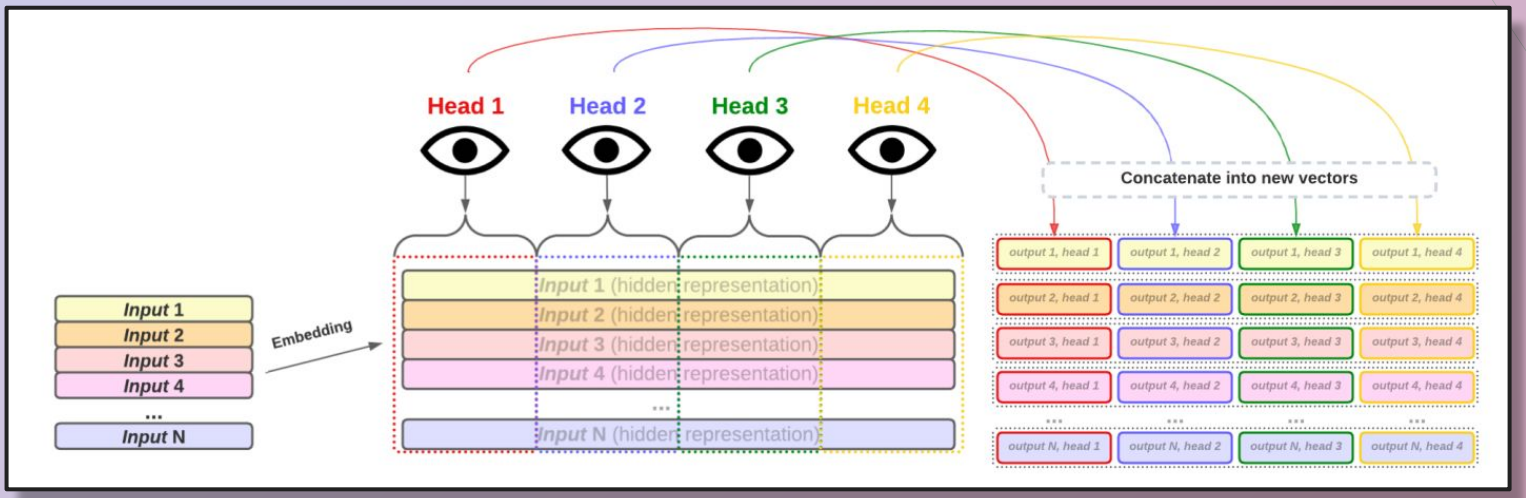
Ahora ya tenemos una medida de la importancia entre el nodo A y el nodo vecino i.
Recordemos que esto se calcula automáticamente, gracias a que entrenamos a la red a aprender W_att, y W , que hacen posible asignar *"attention scores"* entre vectores de características.

IDAL Intelligent Data Analysis Laboratory

Fundamento matemático en GANs

En la práctica todo se complica un poquito más porque los mecanismos de atención no se implementan sobre el vector oculto completo (h_i, h_A), sino que se hace mediante el famoso método ***Multi-head attention.***

El motivo, del cual ya se dieron cuenta en el artículo original donde se exponían los transformers por primera vez, es que la implementación anterior no es muy estable.



IDAL Intelligent Data Analysis Laboratory

Fundamento matemático en GANs

El proceso es: Calcular los vectores de representación intermedia. Separarlos en partes iguales y aplicar atención sobre cada uno de ellos, y por último concatenarlos o también se propone la opción de agregarlos .

$$h_i = \|_{k=1}^{n} h_i^k = \|_{k=1}^{n} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k x_j$$

$$h_i = \frac{1}{n} \sum_{k=1}^{n} h_i^k = \frac{1}{n} \sum_{k=1}^{n} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k x_j$$

IDAL Intelligent Data Analysis Laboratory

Fundamento matemático en GANs

Por último, el cálculo vectorial que realizamos es así:

$$H = \tilde{A}^T \boldsymbol{\alpha} X \boldsymbol{W}^T$$

Donde **alpha** es la matriz de atenciones entre nodos, la ponderación de relaciones de importancia (sea como sea que se ha calculado, self-attention simple, multi-head attention, etc.)

Fundamento matemático en GANs

Mejora sobre los mecanismos de atención en Graph Attention Networks.

Brody et al. (2021) señalaron que la capa de atención en grafos (GAT) solo calcula un tipo de atención estática, lo cual es un problema para ciertos problemas de grafos. Para solucionar esto, introdujeron una versión mejorada llamada **GATv2**, que calcula una atención dinámica más expresiva. La solución consiste en modificar el orden de las operaciones: la matriz de pesos se aplica después de la concatenación y la matriz de pesos de atención se aplica después de la función de atención:

1. Concatenación de características.
2. Aplicación de la matriz de pesos.
3. Cálculo de la atención estática.

GATv2 introduce una atención dinámica más expresiva para superar las limitaciones del GAT original.

https://arxiv.org/abs/2105.14491

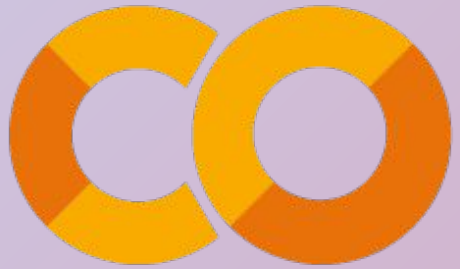
Prácticas

Veamos su aplicación en código

Práctica sobre GAN

Colab Notebook:

https://colab.research.google.com/drive/1aWFQvcmQZJLHF3A-9Nk7BIla1mmbI7MC?usp=sharing



Tecnicas avanzadas

Cosas un poquito más complejas sobre las redes de grafos

GraphSAGE

Escalabilidad en GNNs

IDAL
Intelligent
Data
Analysis
Laboratory

¿Qué es GraphSAGE?

La escalabilidad es crucial cuando se trata de llevar algo a producción.

Hasta ahora, para la forma de trabajar que tienen las **GCNs** y las **GANs**, manejar este **crecimiento desmesurado de los grafos**, supondría un **problema**.

Empresas como Uber Eats y Pinterest adoptaron estas arquitecturas por motivos como la escalabilidad y la eficiencia.

Neighbor sampling

Aggregation operators (node embeddings)

IDAL Intelligent Data Analysis Laboratory

Artículo original (2017): https://arxiv.org/abs/1706.02216

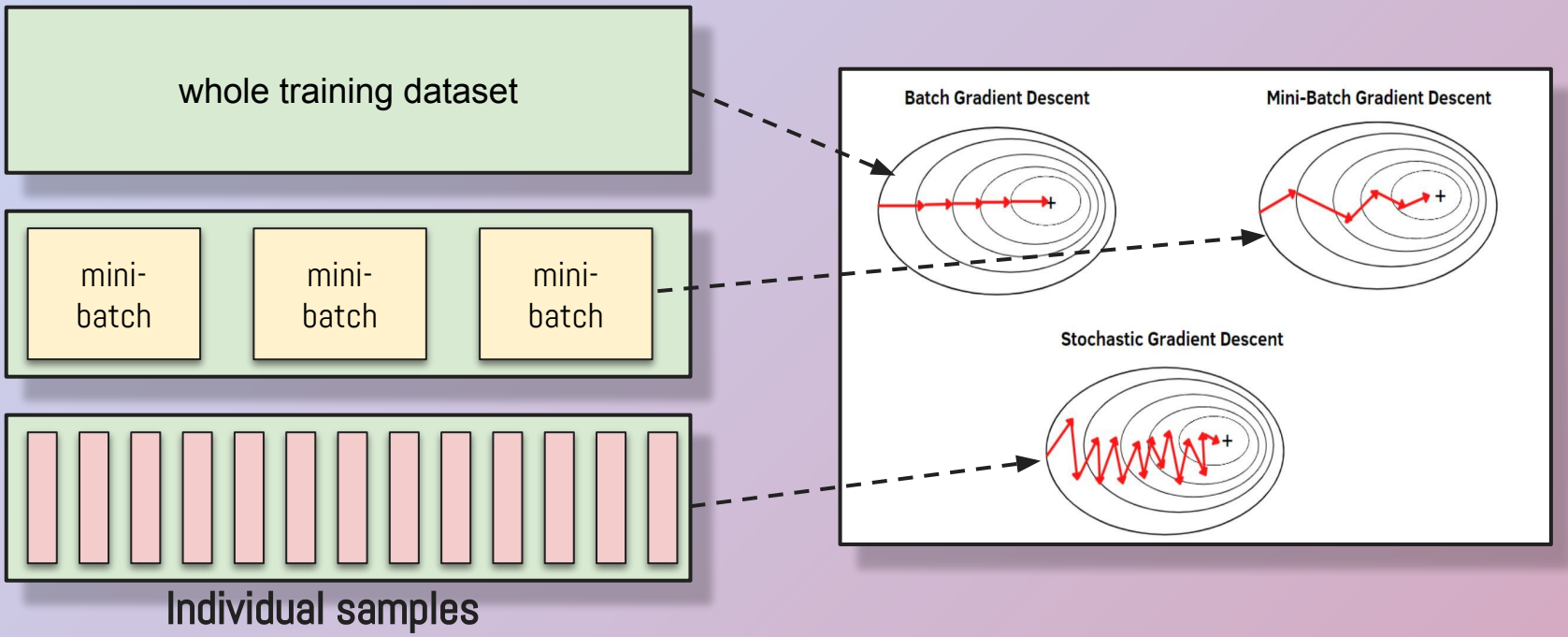
¿Qué es GraphSAGE?

GCNs y las **GANs** presentaba un problema en el escalado y en la generalización a datos nunca vistos.

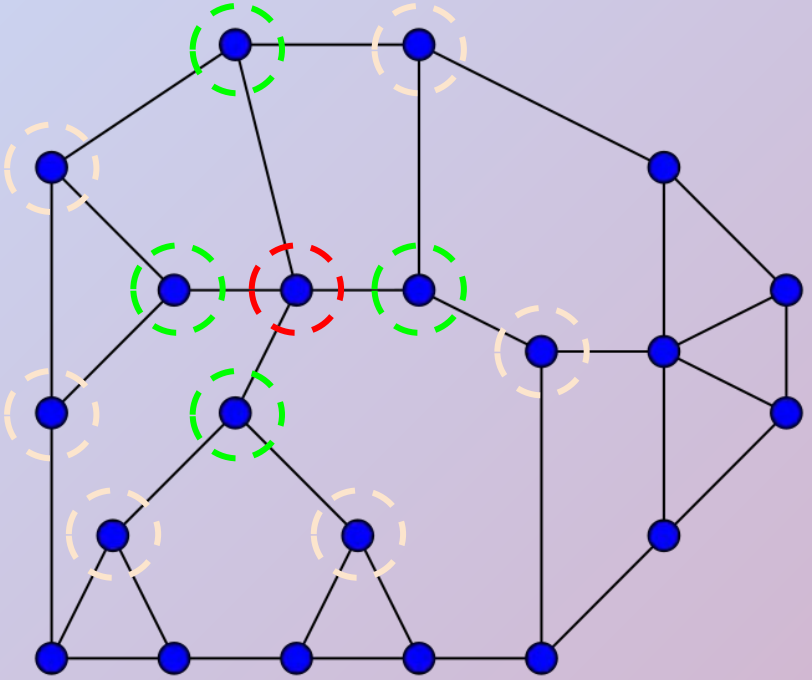
Para resolver este problema, nos preguntamos dos cosas:

- ¿Cómo se haría mini-batching en GNNs?
- ¿Es necesaria TODA la información que usamos en el entrenamiento?

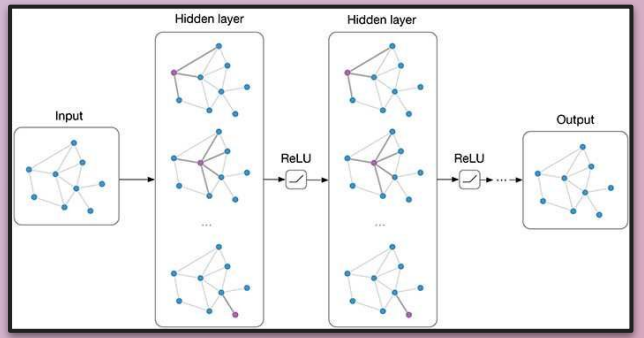
Mini-batching



Mini-batching en GNNs



Según la profundidad de la red, la cantidad de nodos que debemos añadir obligatoriamente en cada mini-batch, crece exponencialmente.



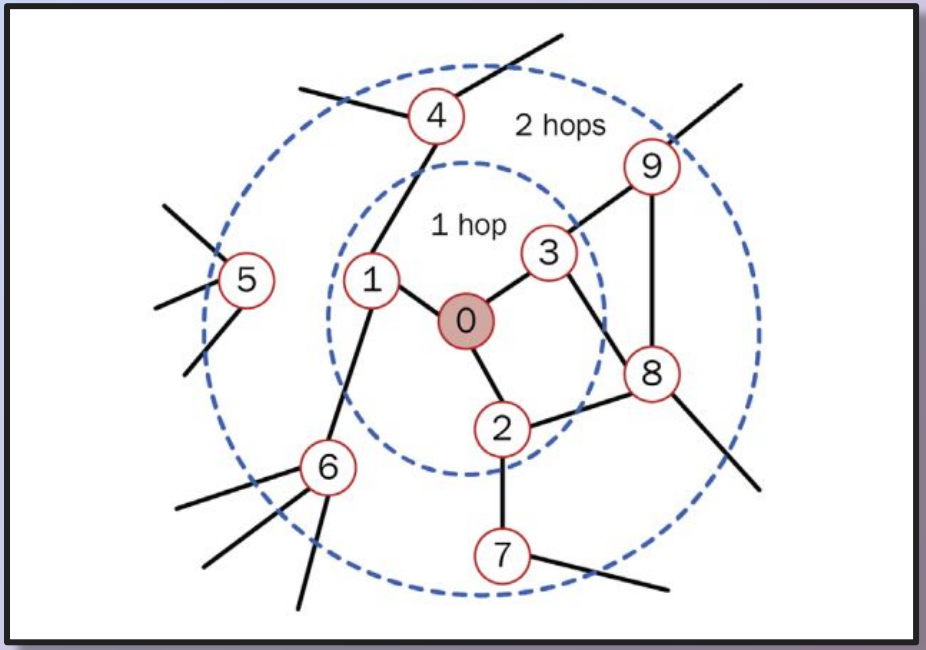
IDAL
Intelligent Data Analysis Laboratory

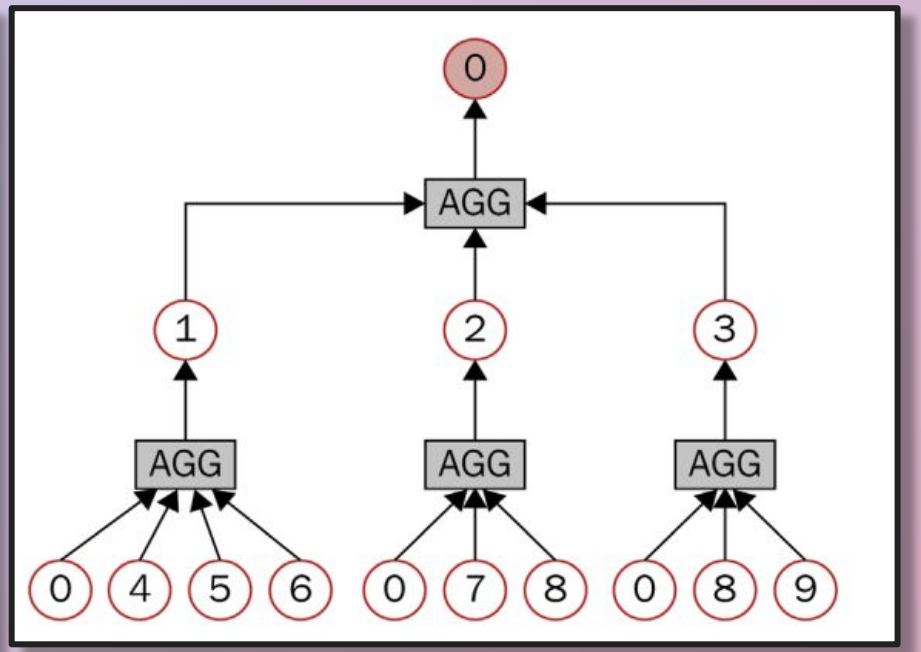
Mini-batching en GNNs

Y hay otro problema, si hubiera **nodos con muchísimas conexiones** (como una celebridad en una red social), se saturaría la estrategia de mini-batching. A estos nodos con una cantidad anormal de conexiones se les conoce como **hub-nodes.**

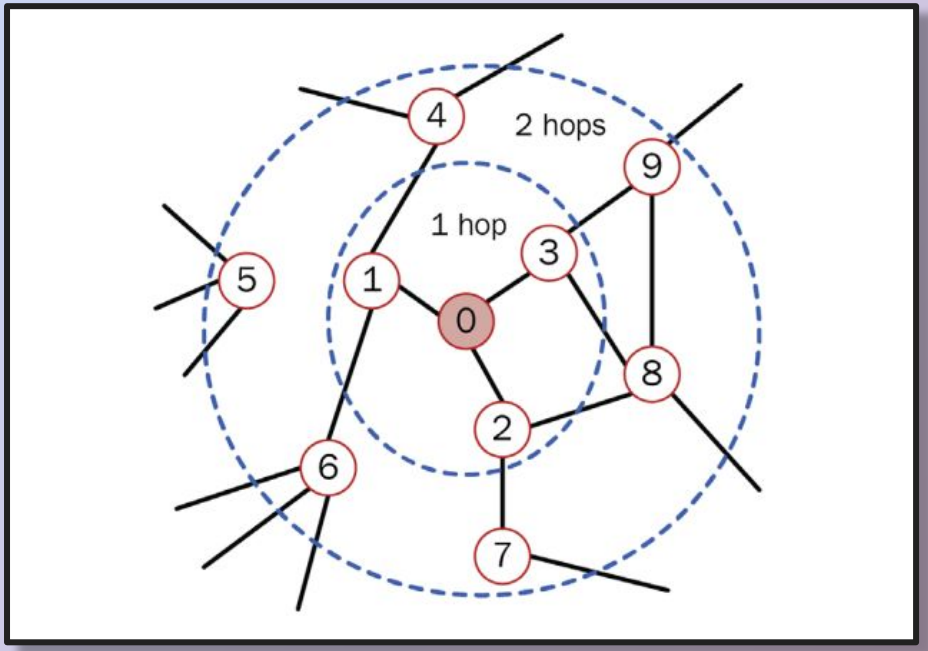
Los autores propusieron una forma de limitar este crecimiento desmesurado, mediante lo que llamaron **neighbor sampling**. En lugar de añadir todos los nodos a la lista de nodos necesarios para el cálculo, se añade solo una cantidad fija, determinada y relativamente pequeña.

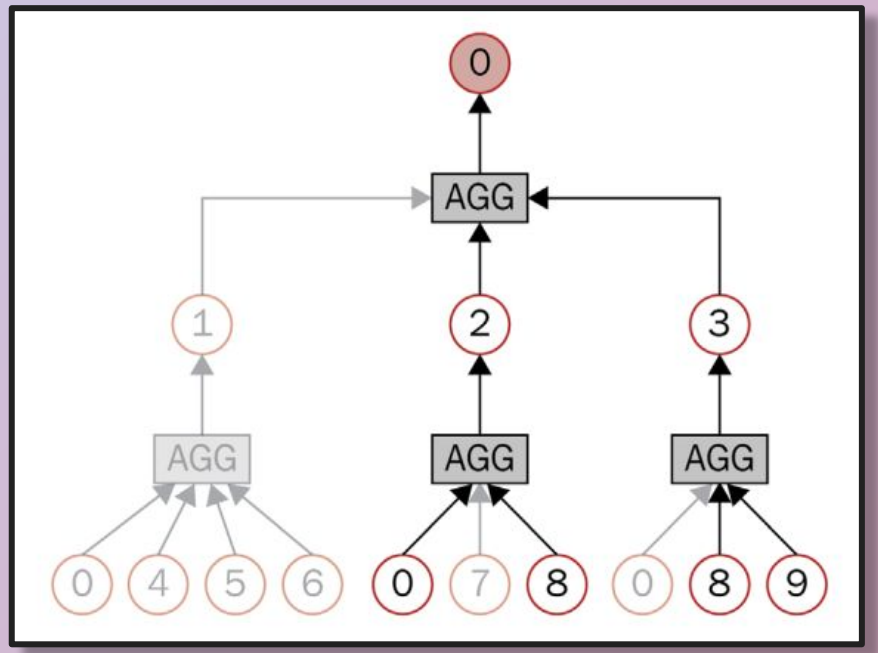
Mini-batching en GNNs





Neighbor Sampling





IDAL Intelligent Data Analysis Laboratory

Se toman fijos dos nodos en el **1-hop**, y otros dos nodos en el **2-hop**.

Neighbor Sampling

Hay un equilibrio:

- Por un lado, elegir pocos nodos, es muy óptimo y eficiente, pero poco preciso y añade mucha aleatoriedad al entrenamiento.
- Un número de sampleo nos puede devolver a la casilla de salida, donde no era eficiente ni escalable.

Sobre la estrategia de sampleo hay mucha teoría escrita. Por ejemplo, ***PinSAGE*** es la variante usada por Pinterest para su sistema recomendador. En esta variante, se usan *"random walks"*, para determinar los nodos más frecuentes y poder incluirlos en el sampleo. De modo que en la práctica, no hace falta incluir todos los nodos vecinos, sino sólo los más críticos

PinSAGE (2018): https://arxiv.org/pdf/1806.01973

Aggregation

Ahora que ya se han *"sub-sampleado"* los nodos para el proceso de mini-batching, se plantea como realizar la agregación de estos.

- ***Mean aggregator.***
- ***LSTM aggregator.***
- ***Pooling aggregator.***

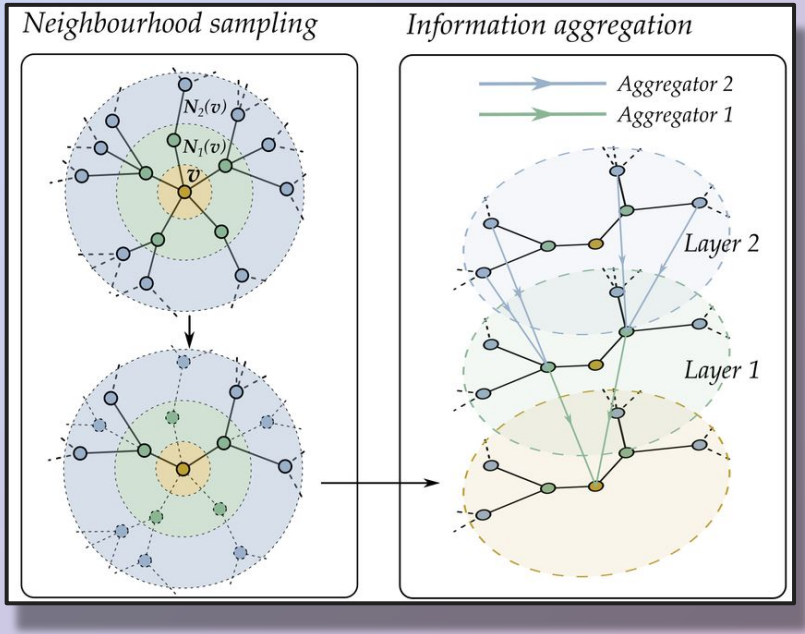
$$h_i^{'} = \sigma\left(\mathbf{W} \cdot mean_{j \in \tilde{\mathcal{N}}_i}(h_j)\right)$$

$$h_i^{'} = \sigma\left(\mathbf{W}_1 h_i + \mathbf{W}_2 \cdot mean_{j \in \mathcal{N}_i}(h_j)\right)$$

IDAL Intelligent Data Analysis Laboratory

Para más información vease el artículo original:
https://cs.stanford.edu/~jure/pubs/graphsage-nips17.pdf

GraphSAGE overview



Message Passing

1. Generación del mensaje

   Cada nodo genera mensajes para sus vecinos usando sus características propias y, opcionalmente, las de los vecinos y las aristas.

2. Agregación del mensaje

   Cada nodo combina mensajes de sus vecinos mediante una función que no cambia con las permutaciones, típicamente sumando estos mensajes

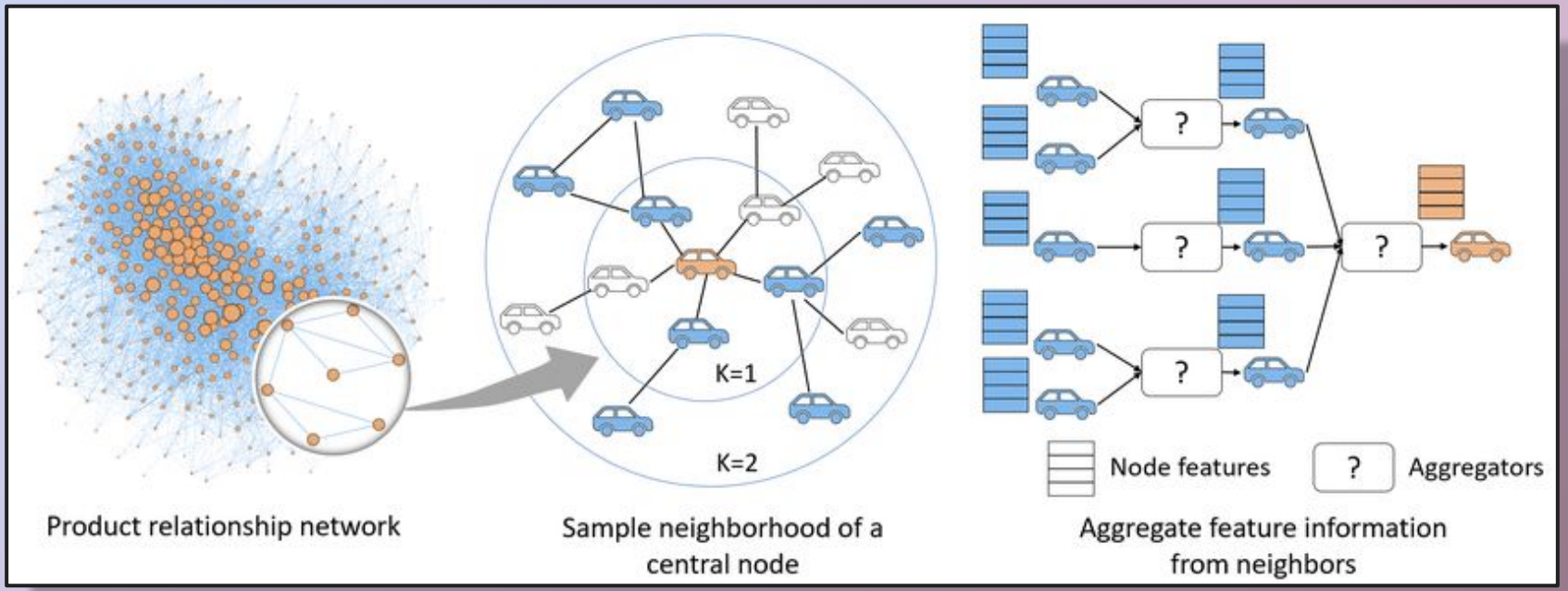
3. Actualización del estado.

   Actualiza sus propias características integrando estos mensajes agregados. Incluyéndose, a veces, el estado propio con los estados de los vecinos.

GraphSAGE entra dentro del ***"framework"*** de ***message passing***, pues estamos generando el mensaje *(neighbour sampling)*, agregando el mensaje *(aggregation)*, y actualizando el estado.

Para más información vease el artículo original:
https://cs.stanford.edu/~jure/pubs/graphsage-nips17.pdf

GraphSAGE overview



Para más información vease el artículo original:
https://cs.stanford.edu/~jure/pubs/graphsage-nips17.pdf

IDAL Intelligent Data Analysis Laboratory

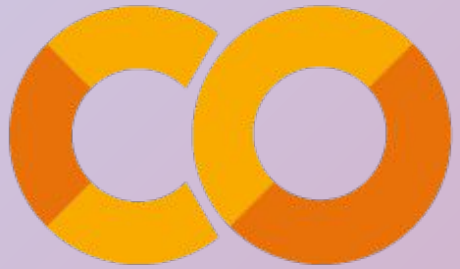
Prácticas

Veamos su aplicación en código

Práctica sobre GraphSAGE

Colab Notebook:

https://colab.research.google.com/drive/1qbQSTIcqkqNXG8T99Mve__Ek7BKI2Gnm?usp=sharing



IDAL Intelligent Data Analysis Laboratory

Más aplicaciones prácticas

De qué otras formas se plantea el uso de redes neuronales de grafos en la práctica.

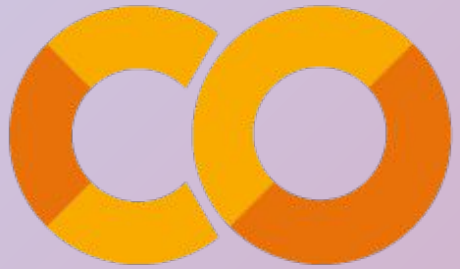
Prácticas

Veamos su aplicación en código

Práctica sobre Aplicaciones GNNs

Colab Notebook:

https://colab.research.google.com/drive/1r31pG3cKmxCeSw5NfnZWQT0CAAv__Q2k?usp=sharing



IDAL Intelligent Data Analysis Laboratory

Gracias por vuestra atención

¿ Preguntas ?

IDAL Intelligent Data Analysis Laboratory

LinkedIn: Arturo Sirvent Fresneda