

# GRNNs

## Introducción a las redes neuronales de Grafos

---

Seminario máster

16 OCT 2024

Arturo Sirvent Fresneda

# Sobre mi

---

- Soy de **Alicante** ciudad.
- Estudié **física en la Universidad de Granada**, pero me especialicé en física computacional y análisis de datos.
- Hice el máster en **Ciencia de Datos** en la UV (ETSE).
- He trabajado como **Ingeniero de Inteligencia Artificial** en el proyecto **SmartCity Valencia**, y en la empresa valenciana **Tyris AI**, y como **técnico superior de investigación en el Departamento de Ingeniería Mecánica y Biomecánica de la UPV**
- Actualmente estoy trabajando en el instituto universitario **VRAIN (UPV)**.



# Índice seminario

## **Primera parte** (2.5 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

## **Segunda parte** (2.5 horas)

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

# Índice

---

1

## Teoría de Grafos

- ¿Qué partes tiene un grafo?
- ¿Qué tipos hay?
- Fundamentos básicos



IDAL

magister  
en  
Análisis  
y  
Modelado  
de  
Sistemas

# Índice

1

Teoría de Grafos

- ¿Qué partes tiene un grafo?
- ¿Qué tipos hay?
- Fundamentos básicos

2

Representación  
(del dato al grafo)

- ¿Cómo los interpretamos (grafos)?
- ¿Qué datos se benefician de esta representación?
- ¿Cómo pasamos de los datos al grafo?



IDAL

maestría  
doctorado  
investigación  
innovación

# Índice

1

Teoría de Grafos

- ¿Qué partes tiene un grafo?
- ¿Qué tipos hay?
- Fundamentos básicos

2

Representación  
(del dato al grafo)

- ¿Cómo los interpretamos (grafos)?
- ¿Qué datos se benefician de esta representación?
- ¿Cómo pasamos de los datos al grafo?

3

GNNs  
(redes neuronales de grafos)

- Fundamentos
- Taxonomía (partes/clasificación)
- Entrenamiento



# Índice

1 Teoría de Grafos

- ¿Qué partes tiene un grafo?
- ¿Qué tipos hay?
- Fundamentos

2 Representación  
(del dato al grafo)

- ¿Qué tipos de representación existen (grafos)?
- ¿Por qué son más eficientes que otras representaciones?
- ¿Cuáles son las ventajas y desventajas de los datos al grafo?

3  
(redes neuronales)

- Fundamentos
- Taxonomía (partes/clasificación)
- Entrenamiento

# ¿POR QUÉ GRAFOS?

# Problemas de Grafos

---

¿Por qué usar grafos para **representar datos** ?

1. **Ciencias de la Computación:** Modelar la estructura de programas de computadora para entender cómo interactúan los distintos componentes de un sistema.
2. **Física:** Modelar sistemas físicos y sus interacciones, como la relación entre partículas y sus propiedades.
3. **Biología:** Modelar sistemas biológicos, como las vías metabólicas, en forma de redes interconectadas.
4. **Ciencias Sociales:** Estudiar y entender redes sociales complejas, incluyendo las relaciones entre individuos en una comunidad.
5. **Finanzas:** Analizar tendencias del mercado de valores y las relaciones entre distintos instrumentos financieros.
6. **Ingeniería:** Modelar y analizar sistemas complejos, como redes de transporte y redes eléctricas.

# Problemas de Grafos

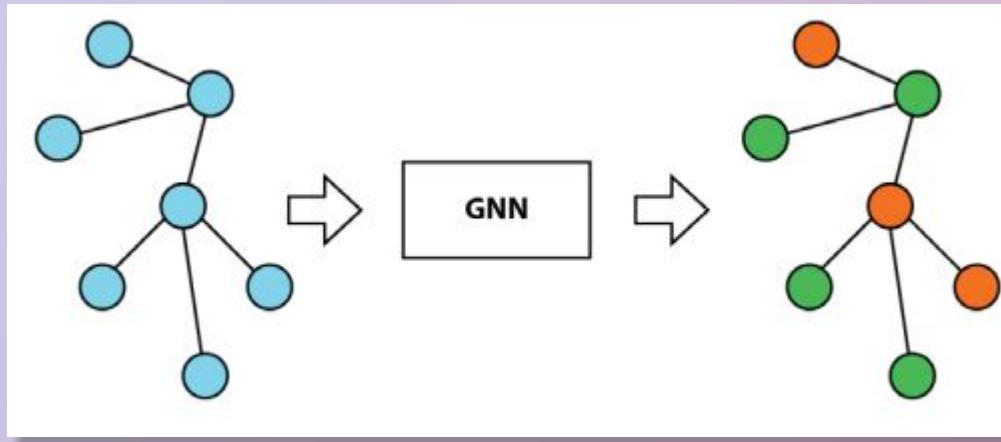
---

¿Por qué usar algoritmos de *machine learning* sobre grafos ?

# Problemas de Grafos

¿Por qué usar algoritmos de *machine learning* sobre grafos?

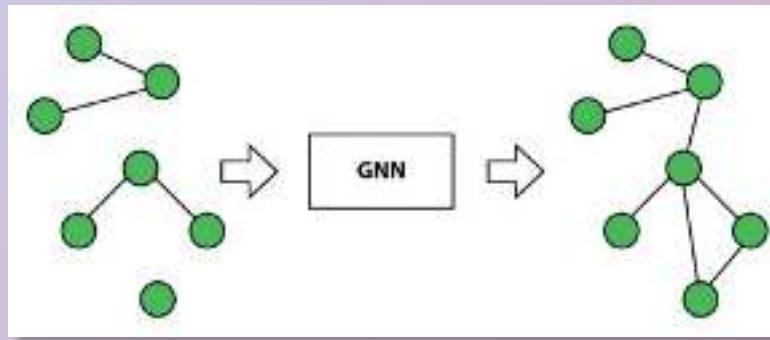
1. **Clasificación de Nodos:** Predecir la categoría de un nodo en un grafo, útil para categorizar usuarios o ítems en línea basándose en sus características.



# Problemas de Grafos

¿Por qué usar algoritmos de *machine learning* sobre grafos?

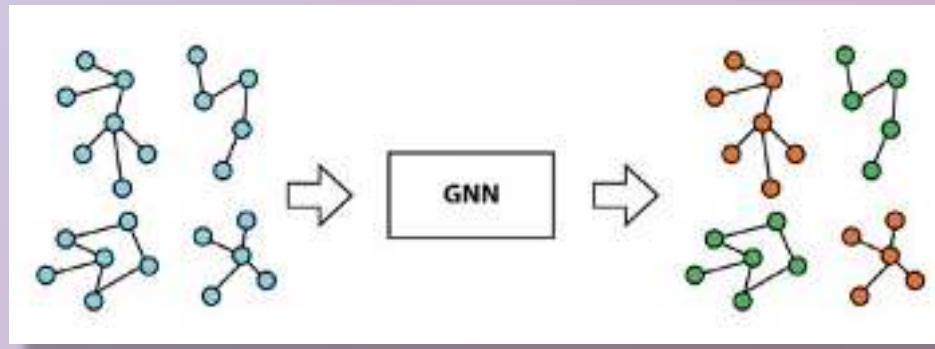
1. **Clasificación de Nodos:** Predecir la categoría de un nodo en un grafo, útil para categorizar usuarios o ítems en línea basándose en sus características.
2. **Predicción de Enlaces:** Predecir enlaces faltantes entre pares de nodos, aplicable en la completación de grafos de conocimiento para prever relaciones, como recomendaciones de amistades (RecSys).



# Problemas de Grafos

¿Por qué usar algoritmos de *machine learning* sobre grafos?

1. **Clasificación de Nodos:** Predecir la categoría de un nodo en un grafo, útil para categorizar usuarios o ítems en línea basándose en sus características.
2. **Predicción de Enlaces:** Predecir enlaces faltantes entre pares de nodos, aplicable en la completación de grafos de conocimiento para prever relaciones, como recomendaciones de amistades (RecSys).
3. **Clasificación de Grafos:** Categorizar diferentes grafos en categorías predefinidas, por ejemplo, para predecir propiedades de estructuras moleculares en biología molecular.



# Problemas de Grafos

---

¿Por qué usar algoritmos de *machine learning* sobre grafos?

1. **Clasificación de Nodos:** Predecir la categoría de un nodo en un grafo, útil para categorizar usuarios o ítems en línea basándose en sus características.
2. **Predicción de Enlaces:** Predecir enlaces faltantes entre pares de nodos, aplicable en la completación de grafos de conocimiento para prever relaciones, como recomendaciones de amistades (RecSys).
3. **Clasificación de Grafos:** Categorizar diferentes grafos en categorías predefinidas, por ejemplo, para predecir propiedades de estructuras moleculares en biología molecular.
4. **Generación de Grafos:** Generar nuevos grafos con propiedades deseadas, como en la creación de estructuras moleculares para descubrimiento de fármacos.

AlphaFold



# Problemas de Grafos

---

¿Por qué usar algoritmos de *machine learning* sobre grafos?

1. **Clasificación de Nodos:** Predecir la categoría de un nodo en un grafo, útil para categorizar usuarios o ítems en línea basándose en sus características.
2. **Predicción de Enlaces:** Predecir enlaces faltantes entre pares de nodos, aplicable en la completación de grafos de conocimiento para prever relaciones, como recomendaciones de amistades (RecSys).
3. **Clasificación de Grafos:** Categorizar diferentes grafos en categorías predefinidas, por ejemplo, para predecir propiedades de estructuras moleculares en biología molecular.
4. **Generación de Grafos:** Generar nuevos grafos con propiedades deseadas, como en la creación de estructuras moleculares para descubrimiento de fármacos.

**Más aplicaciones Prácticas:** Sistemas de recomendación, para sugerir ítems a usuarios basándose en interacciones previas. Predicción del tráfico, mejorando las estimaciones de tiempos de viaje al considerar relaciones complejas entre rutas y modos de transporte ...

# Teoría de Grafos

1

Teoría de Grafos

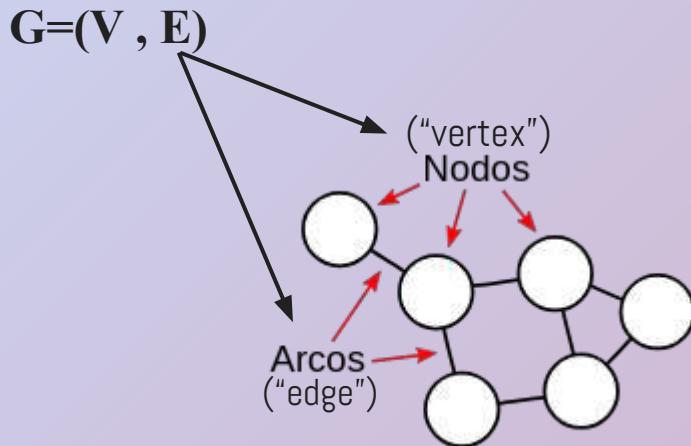
- ¿Qué partes tiene un grafo?
- ¿Qué tipos hay?
- Fundamentos básicos

# Teoría de Grafos

1

## Teoría de Grafos

- ¿Qué partes tiene un grafo?
- ¿Qué tipos hay?
- Fundamentos básicos



Los nodos representan objetos como:

- Ciudades
- Personas
- Páginas web

Los arcos representan las conexiones y relaciones entre nodos.



# Teoría de Grafos

1

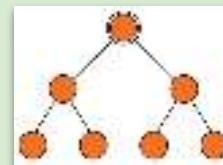
## Teoría de Grafos

- ¿Qué partes tiene un grafo?
- **¿Qué tipos hay?**
- Fundamentos básicos

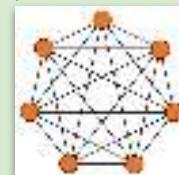
- Dirigido o no dirigido (dirección en conexiones).
- Conexiones ponderadas ("weighted")
- Conexo o inconexo.

### Los más famosos:

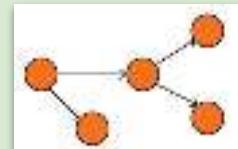
Árbol



Completo



Directed Acyclic Graph (DAG)



# Teoría de Grafos

1

## Teoría de Grafos

- ¿Qué partes tiene un grafo?
- ¿Qué tipos hay?
- Fundamentos básicos

### Términos:

- **Grado de un nodo** : Número de conexiones que un nodo tiene.
- **Vecinos:** Nodos conectados por un solo arco (grado 1).
- **Nodos adyacentes:** Si tienen un vecino común.
- **Camino ó "path":** Conjunto de arcos que conectan dos nodos.
  - **Sencillo:** No se visitan los nodos más de una vez.
  - **Cíclico:** El primero y último nodo son el mismo. (Acíclico es cuando NO hay ciclos)

# Teoría de Grafos

1

## Teoría de Grafos

- ¿Qué partes tiene un grafo?
- ¿Qué tipos hay?
- Fundamentos básicos

### Métricas :

- **"Graph's minimum cut "**: Mínimo número de conexiones a eliminar para hacerlo inconexo. Se usa mucho en problemas de optimización de flujos, agrupación, detección de comunidades.
- **Grado de centralidad ( *Degree Centrality* )**: Grado del nodo, es decir, el número de conexiones directas que el nodo tiene con otros nodos.
- **Centralidad de Cercanía ( *Closeness Centrality* )**: Qué tan cerca está un nodo de todos los demás nodos en el grafo, correspondiendo a la longitud promedio del camino más corto entre el nodo objetivo y todos los otros nodos en el grafo.
- **Centralidad de Intermediación ( *Betweenness Centrality* )**: Mide el número de veces que un nodo aparece en el camino más corto entre pares de otros nodos en el grafo.
- **Densidad:** Indica cuán conectado está un grafo. Se trata de una relación entre el número real de aristas y el número máximo posible de aristas en el grafo.

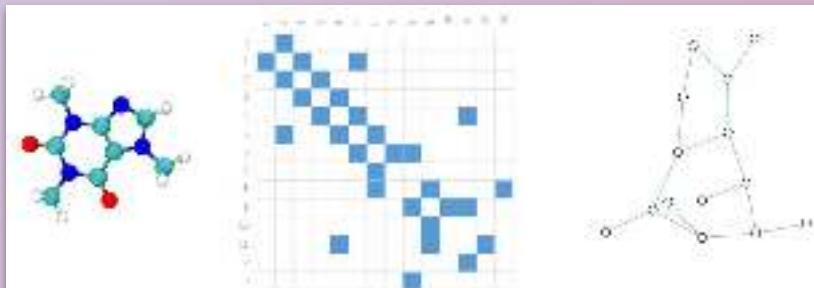
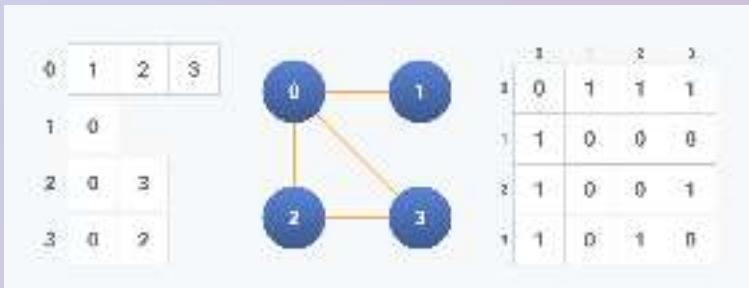
# Representación de Grafos

2

Representación  
(del dato al grafo)

- ¿Cómo los interpretamos (grafos)?
- ¿Qué datos se benefician de esta representación?
- ¿Cómo pasamos de los datos al grafo?

Según el caso concreto convendrá más una representación u otra. La **representación más habitual**, es mediante la **matriz de adyacencia**:

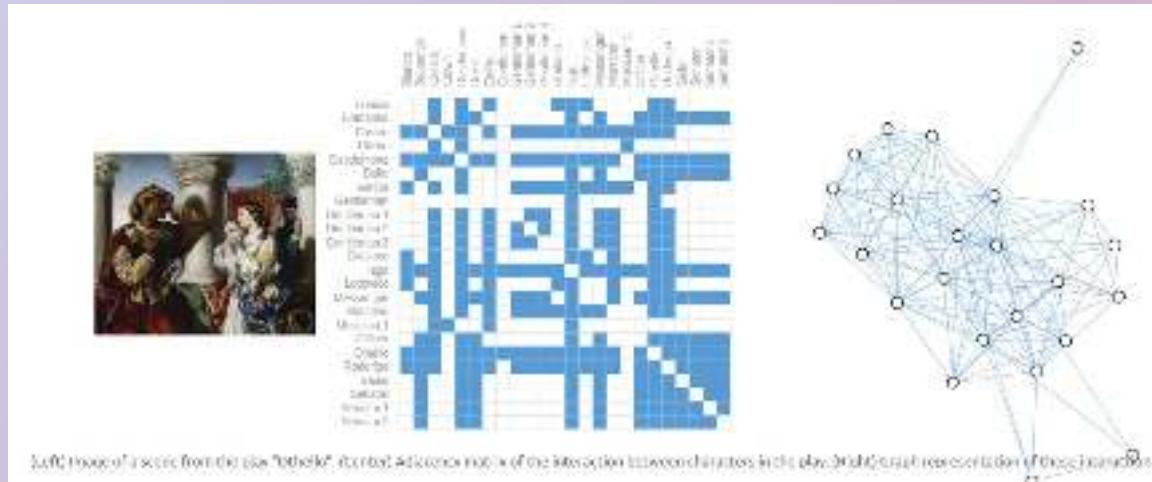


# Representación de Grafos

2

Representación  
(del dato al grafo)

- ¿Cómo los interpretamos (grafos)?
- **¿Qué datos se benefician de esta representación?**
- ¿Cómo pasamos de los datos al grafo?



Left: Image of a scene from the play "Romeo and Juliet". Right: A network graph showing the interaction between characters in the play. 3D: Graph representation of the characters in the play.

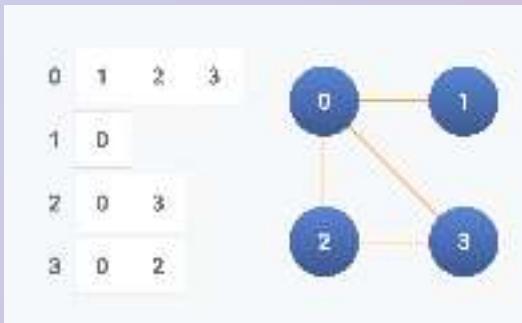
# Representación de Grafos

2

Representación  
(del dato al grafo)

- ¿Cómo los interpretamos (grafos)?
- ¿Qué datos se benefician de esta representación?
- ¿Cómo pasamos de los datos al grafo?

Si tenemos un **grafo muy disperso (sparse)**, donde hay muchos más nodos que conexiones, entonces lo más recomendable es trabajar con **listas de adyacencia**.



Por ejemplo:

- Recomendadores:
  - Gustos musicales en spotify.
  - Follows en Twitter.
  - ...
- Ranking de páginas web (PageRank de Google)

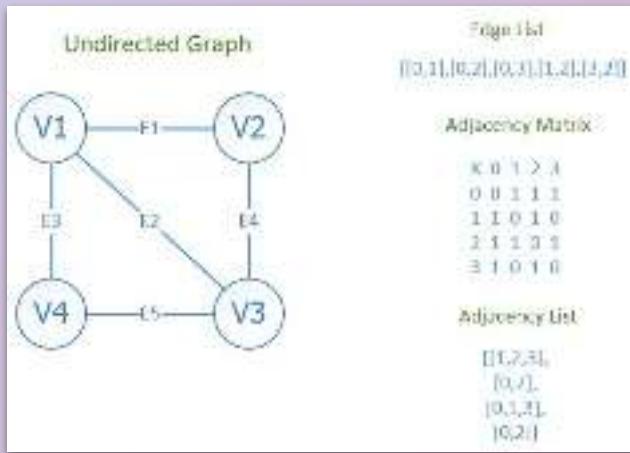
# Representación de Grafos

2

Representación  
(del dato al grafo)

- ¿Cómo los interpretamos (grafos)?
- ¿Qué datos se benefician de esta representación?
- ¿Cómo pasamos de los datos al grafo?

Otra forma de representar estos **grafos dispersos** es mediante las **listas de arcos** (“*edge list*”):



# Representación de Grafos

2

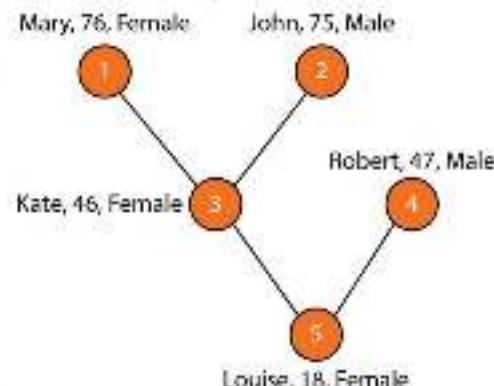
Representación  
(del dato al grafo)

- ¿Cómo los interpretamos (grafos)?
- ¿Qué datos se benefician de esta representación?
- ¿Cómo pasamos de los datos al grafo?

Tabular dataset

ID	Name	Age	Gender
1	Mary	76	Female
2	John	75	Male
3	Kate	46	Female
4	Robert	47	Male
5	Louise	18	Female

Graph dataset



# Representación de Grafos

2

Representación  
(del dato al grafo)

- ¿Cómo los interpretamos (grafos)?
- ¿Qué datos se benefician de esta representación?
- **¿Cómo pasamos de los datos al grafo?**

Unos datos se beneficiarían de la representación en grafo, ¿cómo lo definimos?

- **Definir los Nodos:** Identifica las entidades que serán representadas como nodos en el grafo (ejemplo: usuarios, ciudades, productos).
- **Identificar las Conexiones:** Determina qué tipo de relaciones representarán las aristas entre los nodos (ejemplo: amistades, rutas, transacciones).
- **Crear las Listas de Adyacencia:** Inicia con una lista vacía para cada nodo y añade nodos adyacentes basado en las conexiones reales entre ellos, y de aquí podemos pasar a la matriz de adyacencia.
- **Considerar la Direccionalidad:** Decide si el grafo es dirigido (las relaciones tienen una dirección) o no dirigido (las relaciones son bidireccionales).
- **Ponderar las Aristas (opcional):** Asigna pesos a las aristas si algunas conexiones tienen mayor importancia que otras.

# Prácticas

---

Veamos su aplicación en código

# Práctica sobre la teoría de grafos

---

Colab Notebook:

<https://colab.research.google.com/drive/1T1XRBD8anFx0C12S2gVPC7Mm7qvtTwqG?usp=sharing>



# Algoritmos con grafos

1 Teoría de Grafos

- ¿Qué partes tiene un grafo?
- ¿Qué tipos hay?
- Fundamentos básicos

2 Representación  
(del dato al grafo)

- ¿Cómo los interpretamos (grafos)?
- ¿Qué datos se benefician de esta representación?
- ¿Cómo pasamos de los datos al grafo?

3 GNNs  
(redes neuronales de grafos)

- Fundamentos
- Taxonomía (partes/clasificación)
- Entrenamiento



# Algoritmos con grafos

1

Teoría de Grafos

- ¿Qué partes tiene un grafo?
- ¿Qué tipos hay?
- Fundamentos básicos

2

Representación  
(del dato al grafo)

- ¿Cómo los interpretamos (grafos)?
- ¿Qué datos se benefician de esta representación?
- ¿Cómo pasamos de los datos al grafo?

## Machine Learning PRE-redes neuronales

3

GNNs  
(redes neuronales de grafos)

- Fundamentos
- Taxonomía (partes/clasificación)
- Entrenamiento



# Algoritmos con grafos

## Graph learning techniques

### Graph signal processing

Métodos tradicionales de procesamiento de señales a grafos, como la transformada de Fourier de grafos y el análisis espectral. Revelan propiedades intrínsecas del grafo, como su conectividad y estructura.

# Algoritmos con grafos

## Graph learning techniques

### Graph signal processing

Métodos tradicionales de procesamiento de señales a grafos, como la transformada de Fourier de grafos y el análisis espectral. Revelan propiedades intrínsecas del grafo, como su conectividad y estructura.

### Matrix Factorization

Busca encontrar representaciones de baja dimensión de matrices grandes. Identificar factores latentes o patrones que expliquen las relaciones observadas. Proporcionar una representación compacta e interpretable de los datos.

# Algoritmos con grafos

## Graph learning techniques

### Graph signal processing

Métodos tradicionales de procesamiento de señales a grafos, como la transformada de Fourier de grafos y el análisis espectral. Revelan propiedades intrínsecas del grafo, como su conectividad y estructura.

### Matrix Factorization

Busca encontrar representaciones de baja dimensión de matrices grandes. Identificar factores latentes o patrones que expliquen las relaciones observadas.

Proporcionar una representación compacta e interpretable de los datos.

### Random Walk

Concepto matemático utilizado para modelar el movimiento de entidades en un grafo. Al simular Random Walks sobre un grafo, se puede recopilar información sobre las relaciones entre nodos, lo cual puede dar lugar a "embeddings", representaciones muy útiles para el entrenamiento de otros modelos de aprendizaje automático.



# Algoritmos con grafos

## Graph learning techniques

### Graph signal processing

Métodos tradicionales de procesamiento de señales a grafos, como la transformada de Fourier de grafos y el análisis espectral. Revelan propiedades intrínsecas del grafo, como su conectividad y estructura.

### Matrix Factorization

Busca encontrar representaciones de baja dimensión de matrices grandes. Identificar factores latentes o patrones que expliquen las relaciones observadas. Proporcionar una representación compacta e interpretable de los datos.

### Random Walk

Concepto matemático utilizado para modelar el movimiento de entidades en un grafo. Al simular Random Walks sobre un grafo, se puede recopilar información sobre las relaciones entre nodos, lo cual puede dar lugar a "embeddings", representaciones muy útiles para el entrenamiento de otros modelos de aprendizaje automático.

### Deep Learning

Subcampo del aprendizaje automático centrado en redes neuronales. Muy útiles para codificar, representar y operar eficazmente con datos de grafos como vectores ("embeddings").



# Algoritmos con grafos

## Graph learning techniques

### Graph signal processing

Métodos tradicionales de procesamiento de señales a grafos, como la transformada de Fourier de grafos y el análisis espectral. Revelan propiedades intrínsecas del grafo, como su conectividad y estructura.

### Matrix Factorization

Busca encontrar representaciones de baja dimensión de matrices grandes. Identificar factores latentes o patrones que expliquen las relaciones observadas. Proporcionar una representación compacta e interpretable de los datos.

### Random Walk

Concepto matemático utilizado para modelar el movimiento de entidades en un grafo. Al simular Random Walks sobre un grafo, se puede recopilar información sobre las relaciones entre nodos, lo cual puede dar lugar a "embeddings", representaciones muy útiles para el entrenamiento de otros modelos de aprendizaje automático.

### Deep Learning

Subcampo del aprendizaje automático centrado en redes neuronales. Muy útiles para codificar, representar y operar eficazmente con datos de grafos como vectores ("embeddings").

# Redes neuronales de Grafos

3

GNNs

(redes neuronales de grafos)

- Fundamentos
- Taxonomía (partes/clasificación)
- Entrenamiento

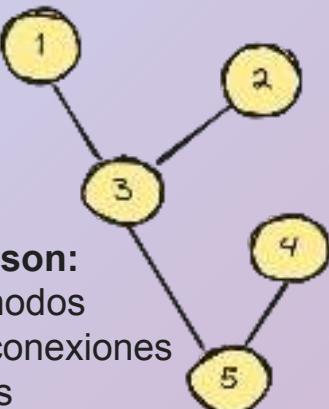
Simplificación del cálculo (hay muchas variantes, por ejemplo, cómo agregar los nodos):

1. Cada nodo tiene unas variables (por ejemplo edad, salario, estado civil, ...)
2. Las relaciones entre nodos también puede llevar consigo variables.
3. Al final, lo que buscamos casi siempre es:
  - a. Clasificación/Regresión de una clase/valor sobre cada nodo.
  - b. Clasificación/Regresión de una clase/valor sobre cada conexión.
  - c. Clasificación/Regresión de una clase/valor sobre el grafo completo.

# Ejemplo cálculos con GNNs

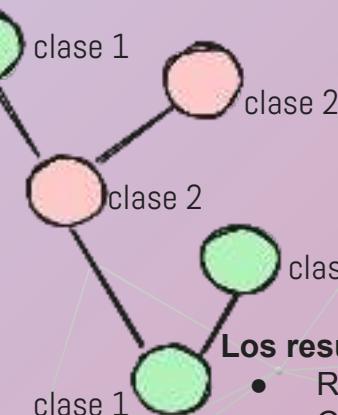
La GNN más sencilla (convolución → GCNs):

- Arquitectura ***graph in - graph out***.
- La conectividad no cambia.
- Message-passing framework



**Los datos de entrada son:**

- Variables de los nodos
- Variables de las conexiones
- Variables globales

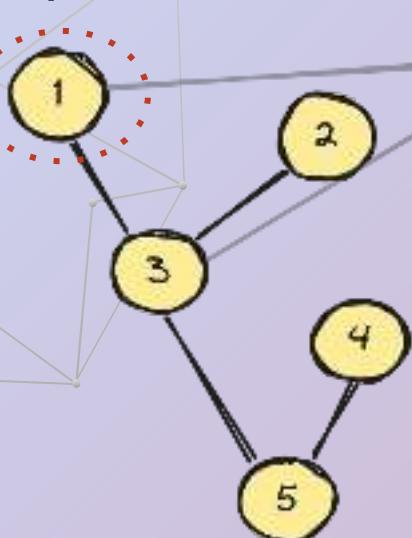


**Los resultados de salida son:**

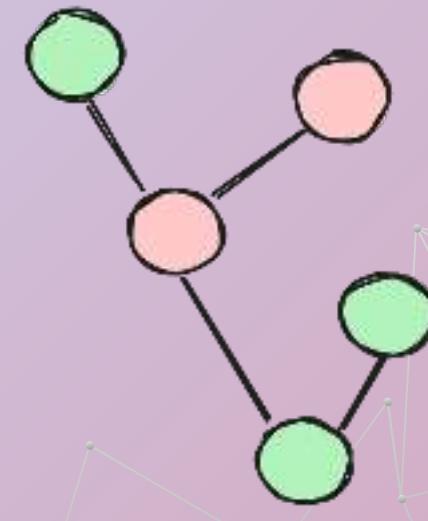
- Regresión sobre nodos
- Clasificación de nodos
- ...

# Ejemplo cálculos con GNNs

Input



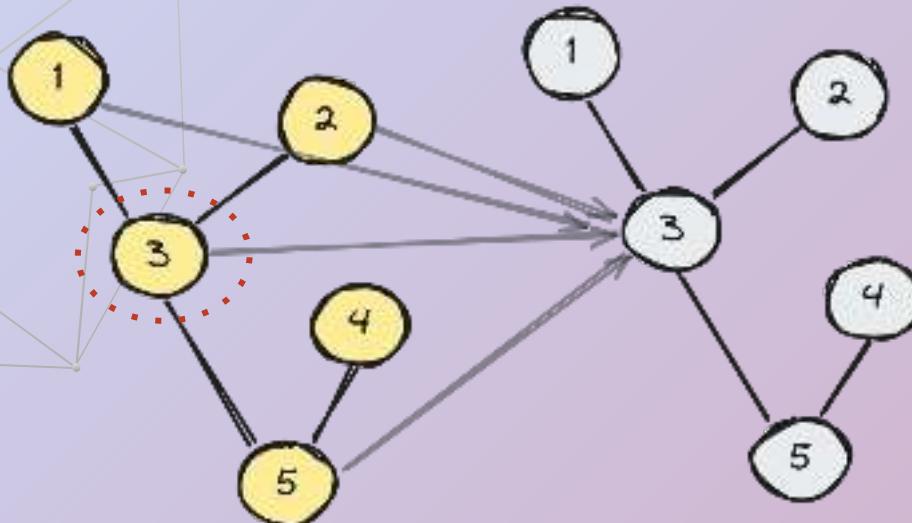
Output



Con 1 capa oculta.



# Ejemplo cálculos con GNNs



$$1 + 3 + 5 + 2$$

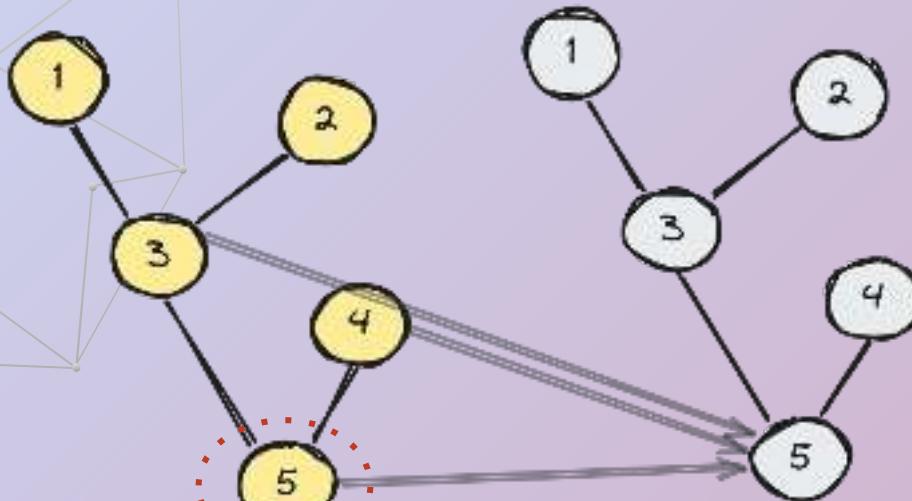


MLP



Con 1 capa oculta.

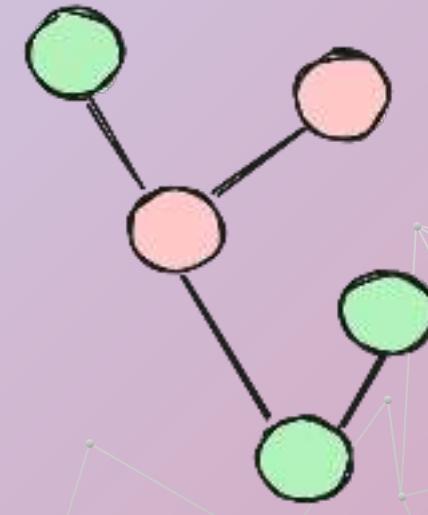
# Ejemplo cálculos con GNNs



MLP

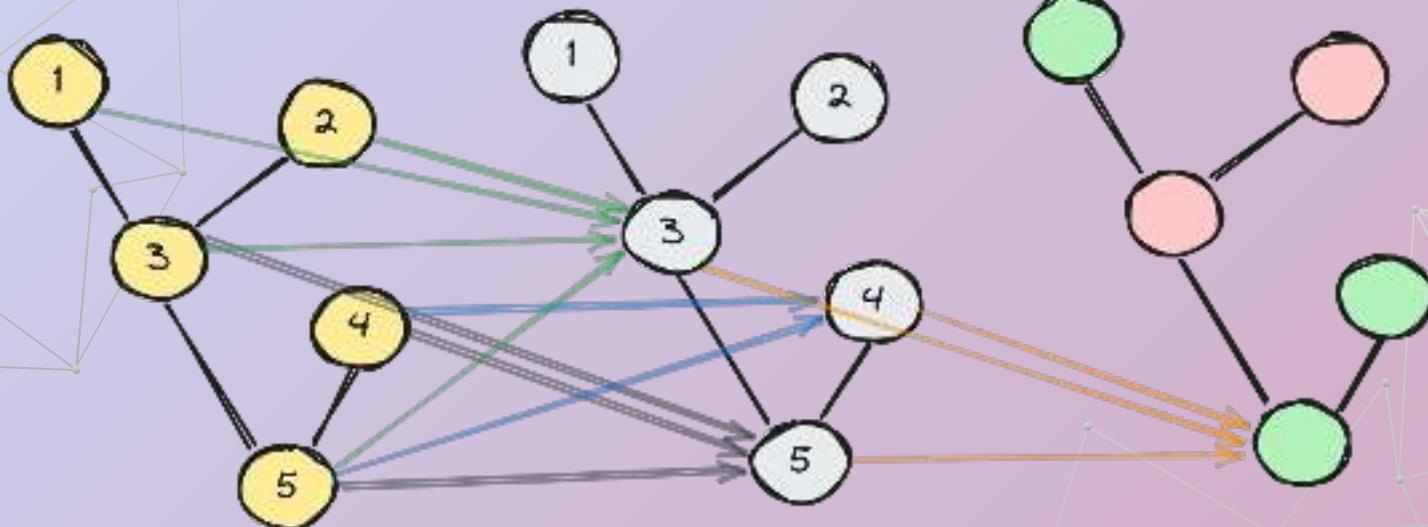
$$4 + 3 + 5 \rightarrow$$

Con 1 capa oculta.



# Ejemplo cálculos con GNNs

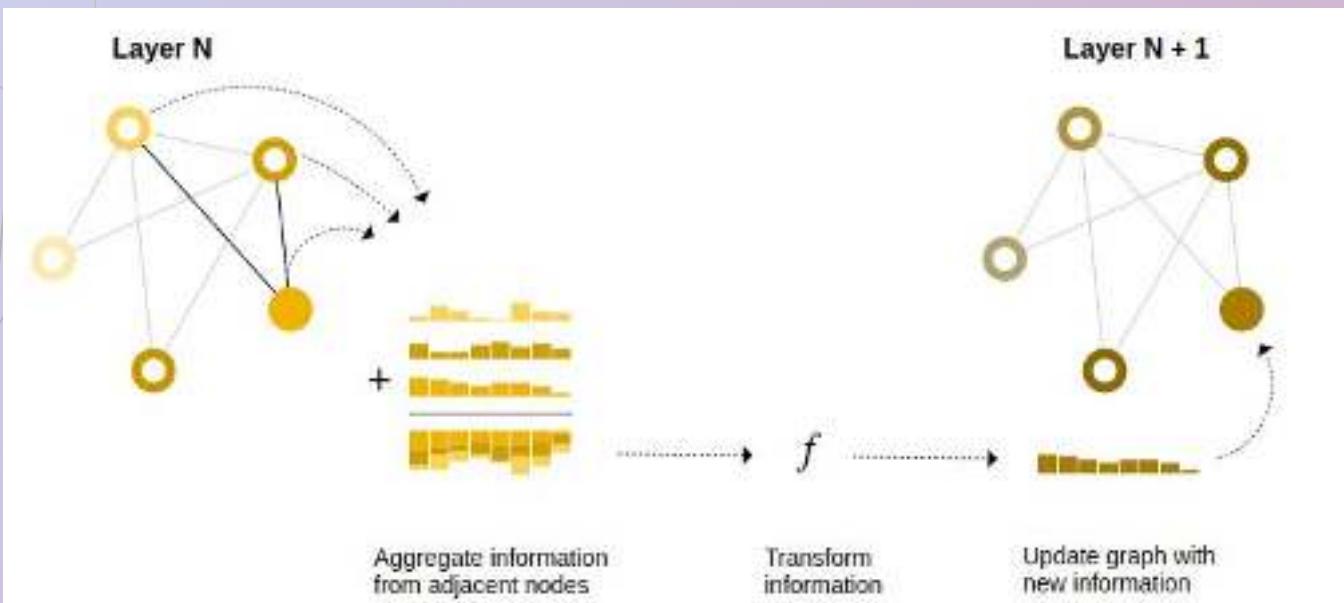
Con 1 capa oculta.



$$4 + 3 + 5 \rightarrow \text{MLP} \rightarrow \text{Output Node}$$

# Ejemplo cálculos con GNNs

Otra visualización:



<https://distill.pub/2021/gnn-intro/>

# Redes neuronales de Grafos

3

GNNs

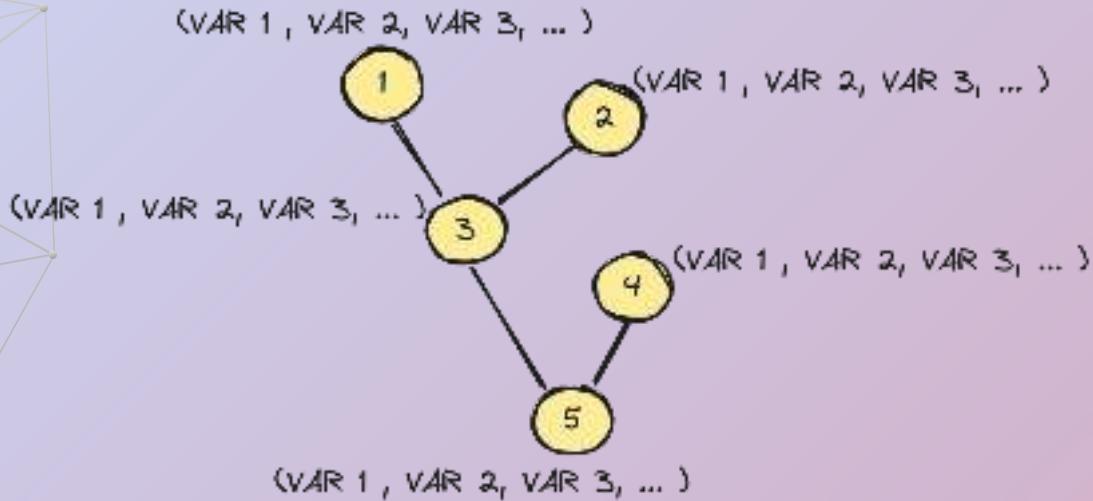
(redes neuronales de grafos)

- Fundamentos
- Taxonomía (partes/clasificación)
- Entrenamiento

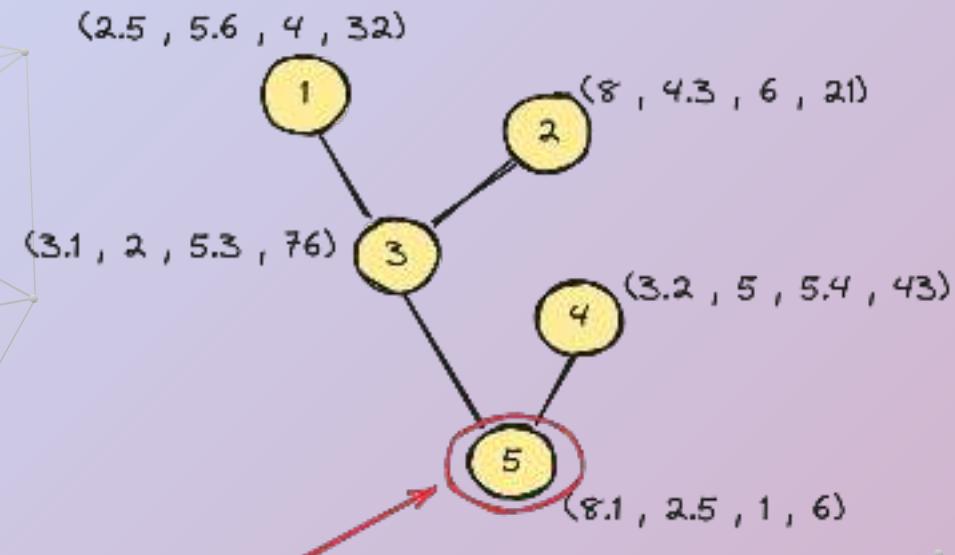
El concepto (framework) de “*message passing*” (2017 by Gilmer et al.). Generalizar las GNNs.



# Ejemplo message passing

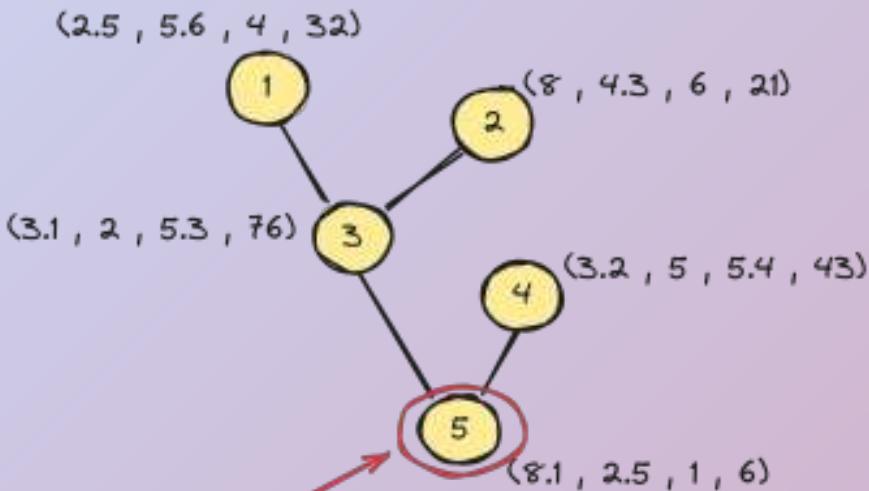


# Ejemplo message passing



Calculemos nueva  
representación de este nodo.

# Ejemplo message passing

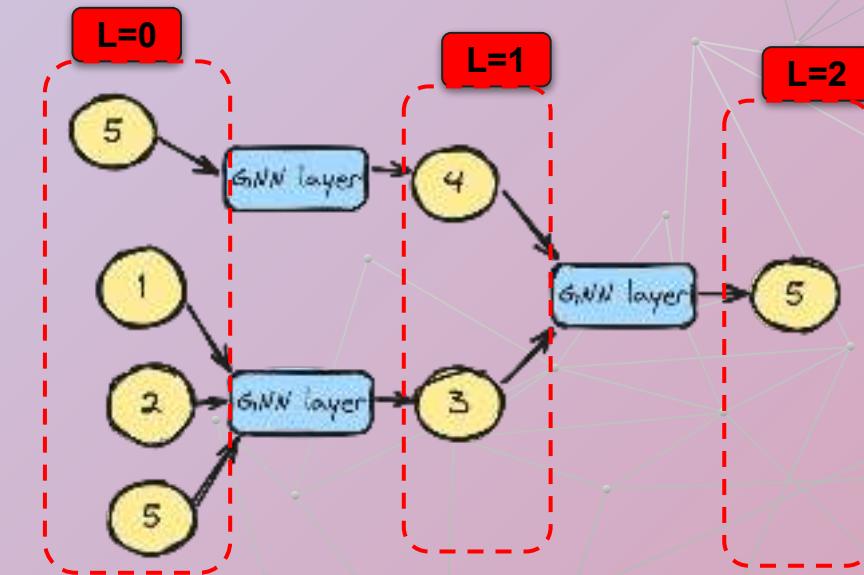
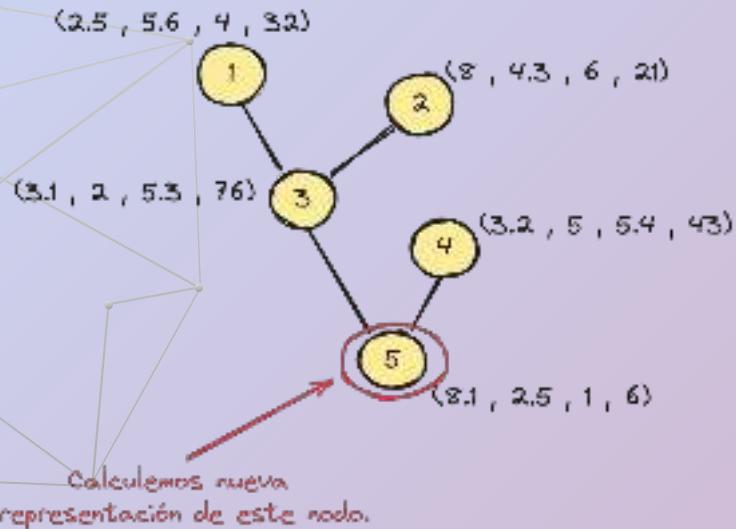


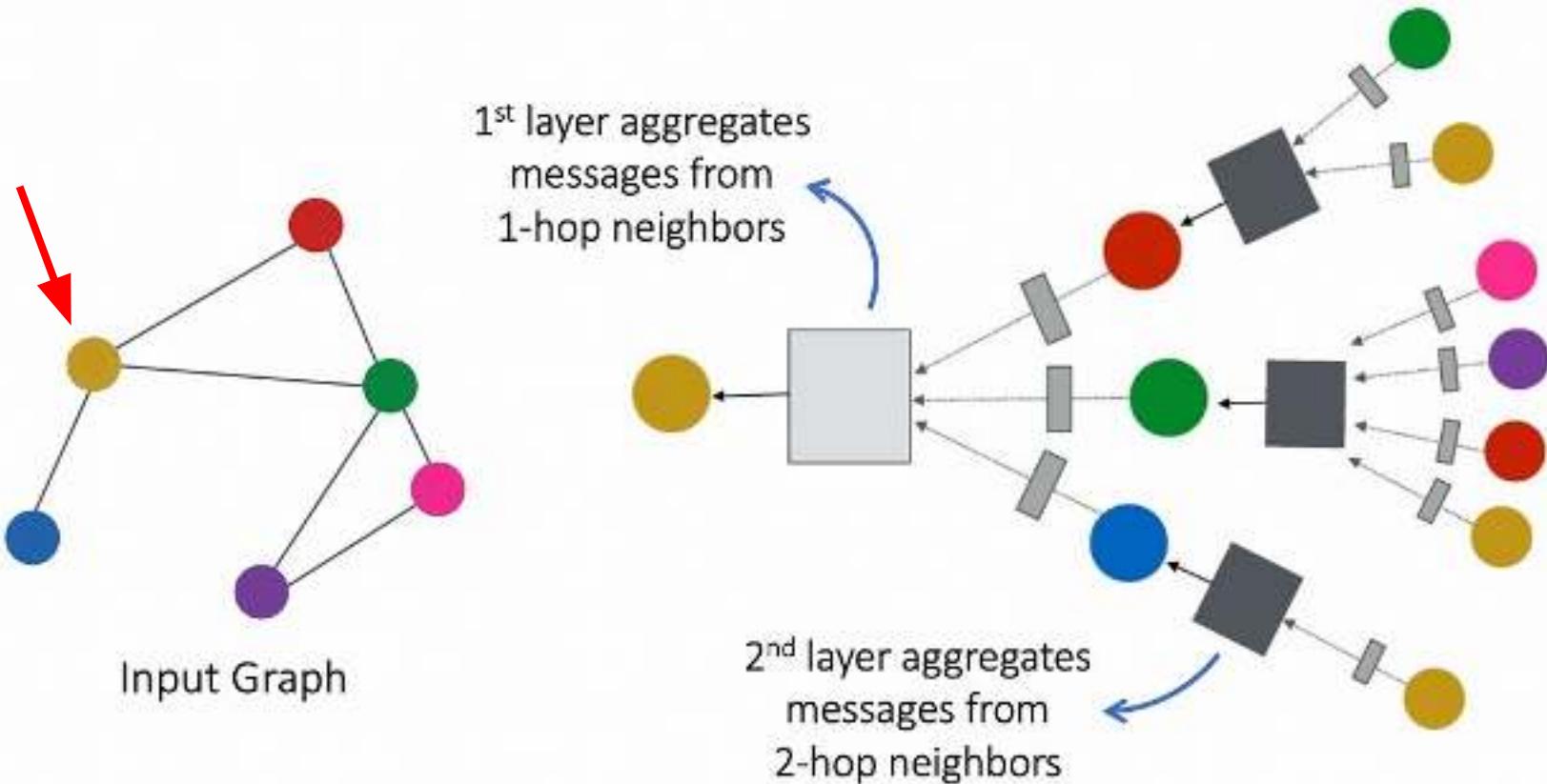
Calculemos nueva  
representación de este nodo.

Para calcular **5** en **L=2** necesitamos **3** y **4** en **L=1** ...

Los cálculos son iterativos, y pueden tener diferente profundidad.

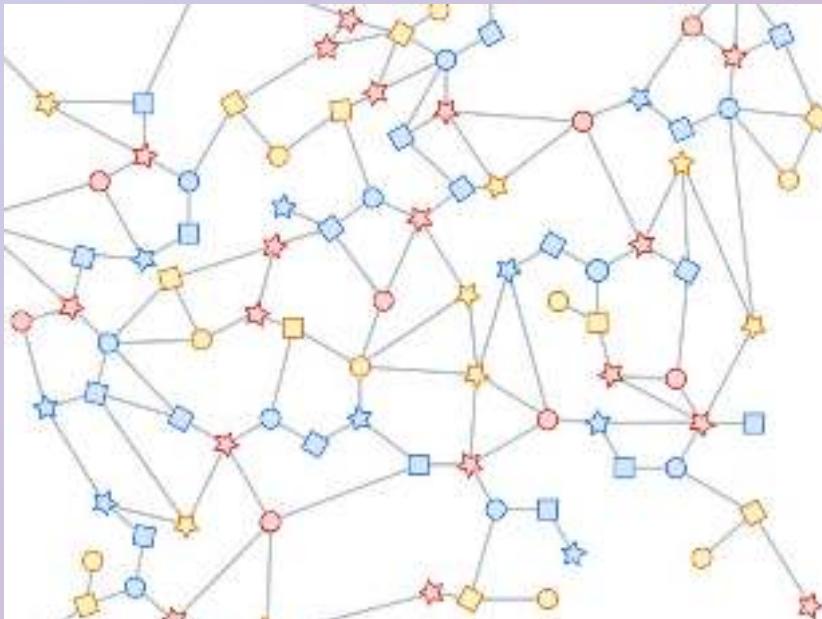
# Ejemplo message passing





# Ejemplo cálculos con GNNs

Otra visualización:



<https://research.google/blog/graph-neural-networks-in-tensorflow/>

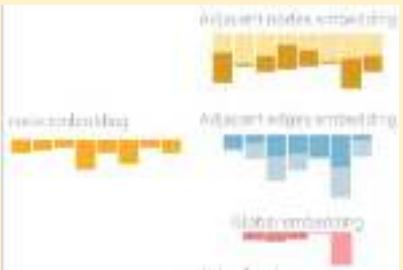
# Ejemplo cálculos con GNNs

Otra visualización:



<https://research.google/blog/graph-neural-networks-in-tensorflow/>

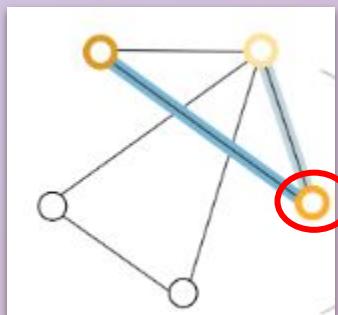
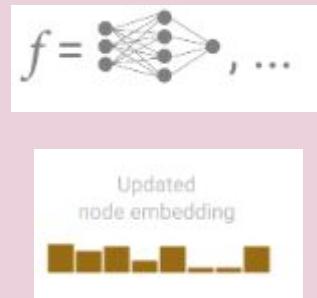
## Mensaje



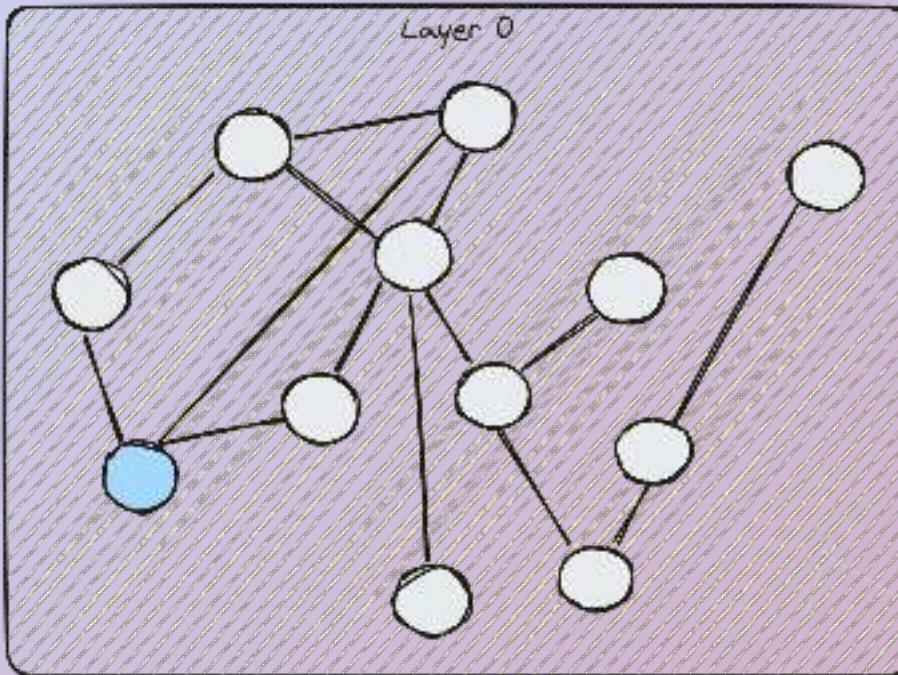
## Agregación

Función invariante a permutaciones.  
E.g. Sum, max, mean,  
...

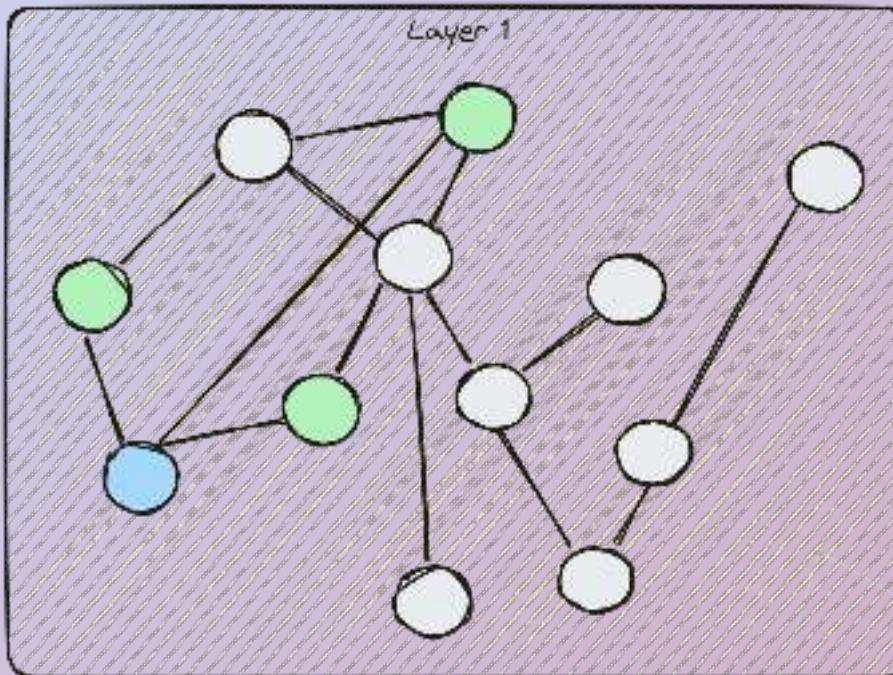
## Actualización



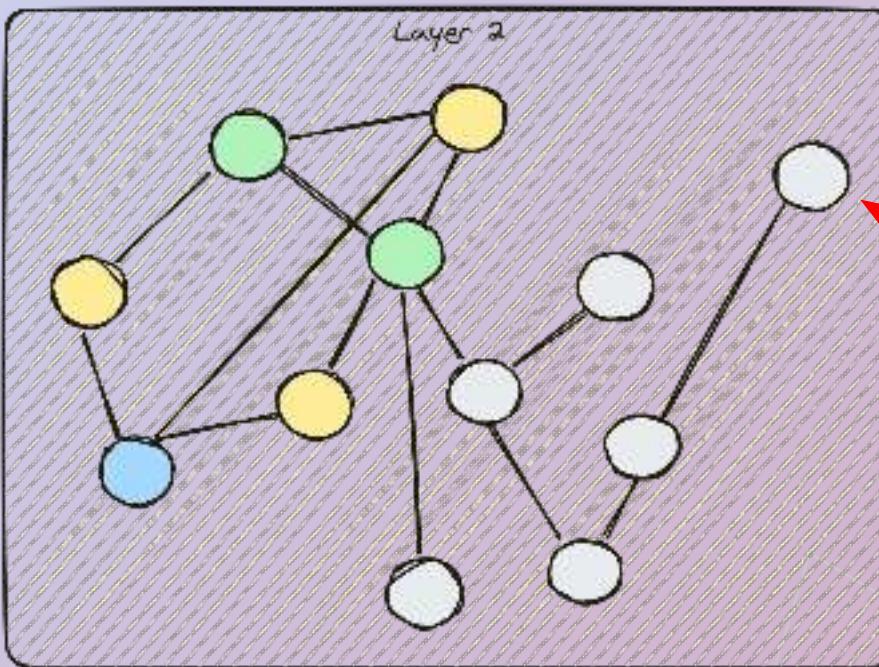
# Alcance de la red ("receptive field")



# Alcance de la red ("receptive field")



# Alcance de la red ("receptive field")



Dificultad para  
abarcar toda la red.

Nota: Se proponen **soluciones** como **nodos maestros** o vectores de contexto ampliado.



# 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**

**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

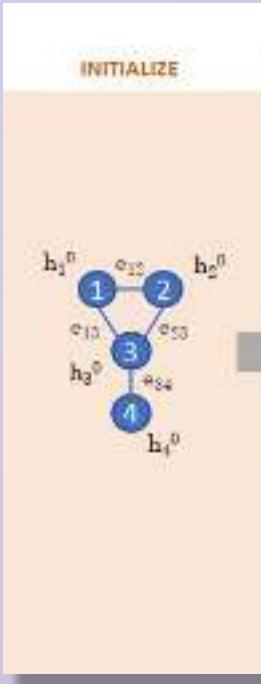
## Message Passing Neural Networks (MPNNs)

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

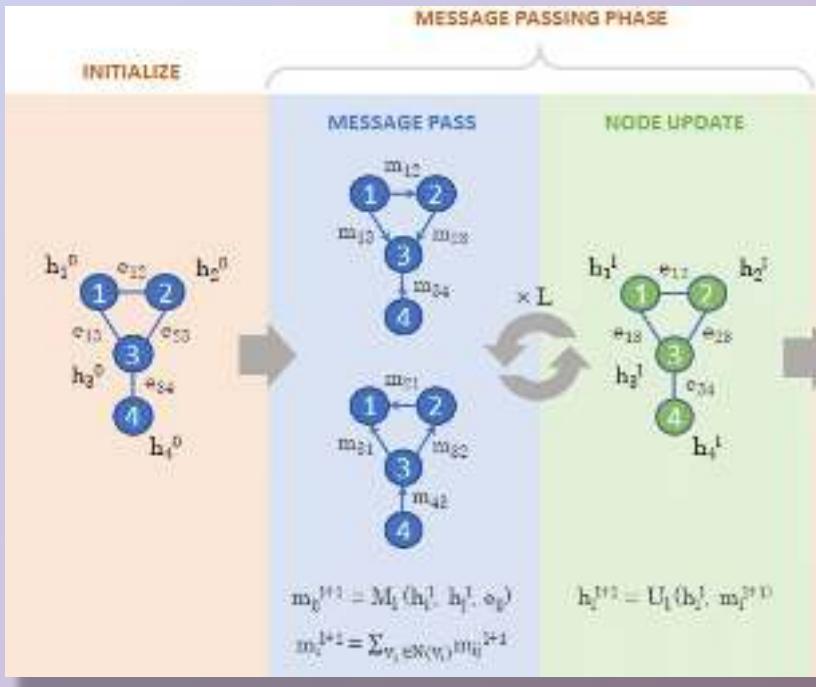
El concepto central es que los **nodos intercambian mensajes con sus vecinos, y estos mensajes son luego agregados y usados para actualizar los estados de los nodos.**

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.

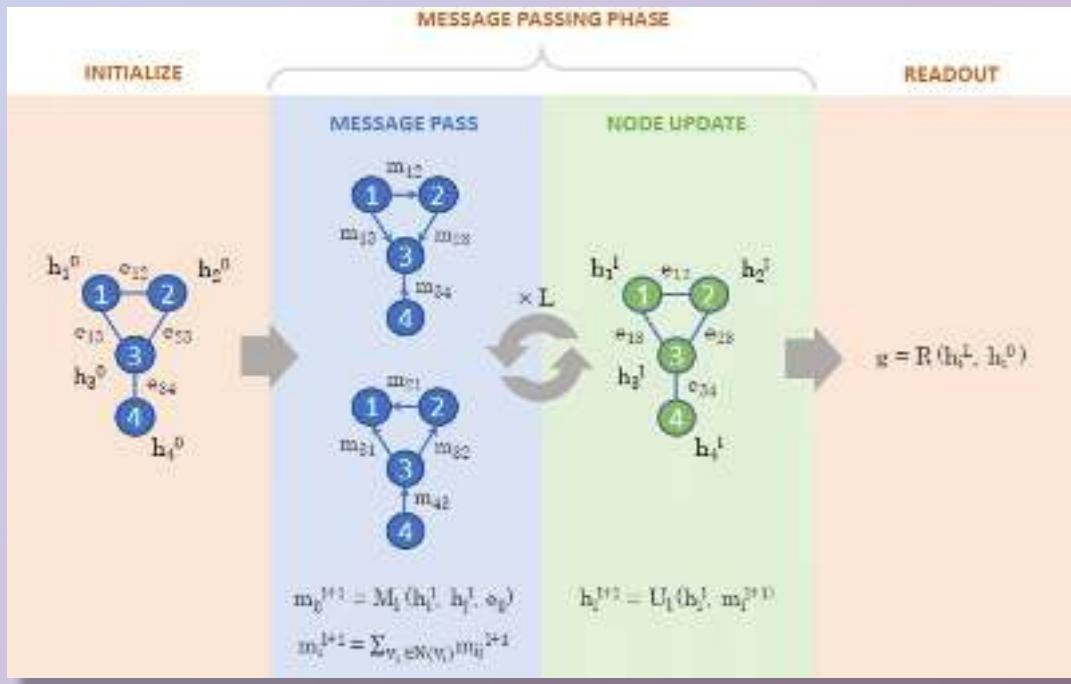
# Redes neuronales de Grafos



# 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.



maestría  
doct.  
investigación  
aprendizaje

IDAL

# 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 (SAmple and aggreGatE):

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, mediante este *sampling*.

# 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, también podría considerarse dentro de dicho marco, pero 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

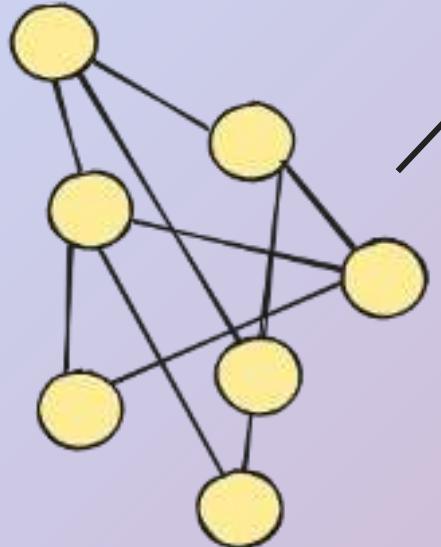
## Redes convolucionales de Grafos (GCNs)

---

¿Cómo trasladar el concepto de convolución a un grafo?

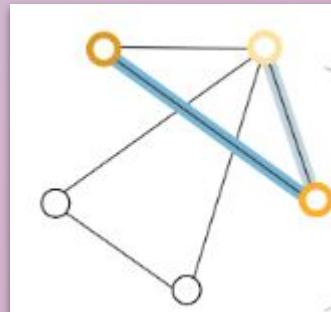
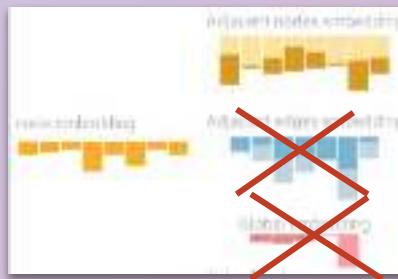
# Fundamento matemático en GCNs

GCN simplificada (“vanilla”)



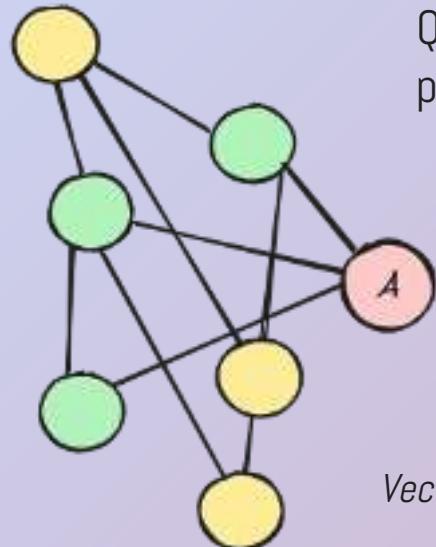
A: Matriz de adyacencia

- 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:

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

Vecinos (nodos verdes)

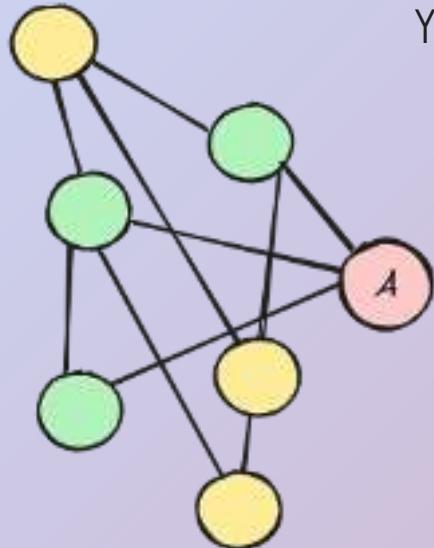
Matriz que transforma linealmente el vector de características

Cada vector de características de los vecinos:

ID	Name	Age	Gender
1	Mary	28	Female
2	John	35	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 .

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

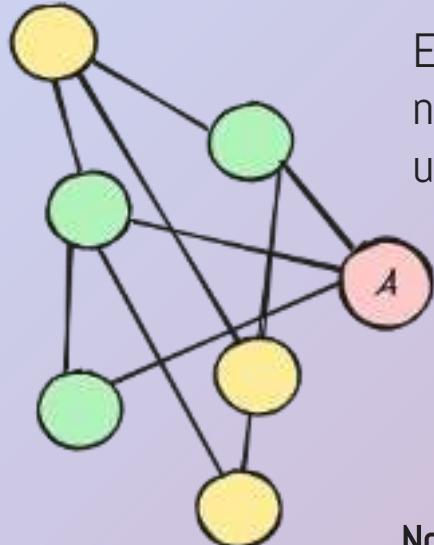
Añadimos la identidad para tener en cuenta el vector de variables del propio nodo a calcular.

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

Aquí estamos calculando todos los estados intermedios  $H=(h_A, h_B, \dots)$  de golpe.

# 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:

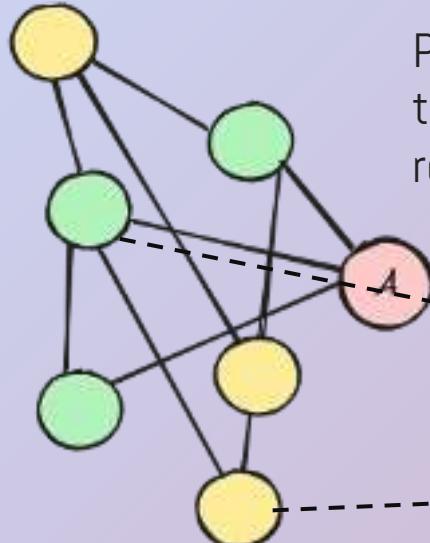
$$h_A = \sum_{i \in N_A} x_i W^T$$

$$h_A = \frac{1}{\deg(A)} \sum_{i \in N_A} x_i W^T$$

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

# Fundamento matemático en GCNs

GCN estandar

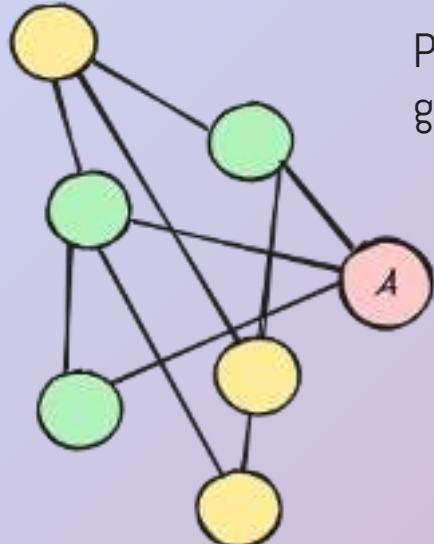


Para hacerlo todo de manera matricial, ahora usaremos también la matriz  $D$  (matrix diagonal de grados), que recoge el grado (número de vecinos) de cada nodo.

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

# Fundamento matemático en GCNs

GCN estandar

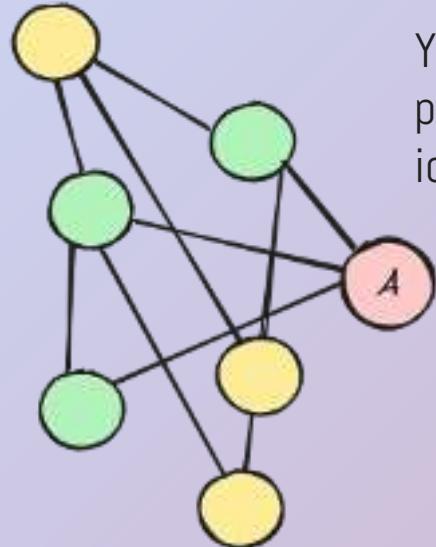


Pero para el cálculo nos interesa  $1/\deg(i)$ , es decir, no el grado, sino su inversa  $1/D$ :

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

# Fundamento matemático en GCNs

GCN estandar



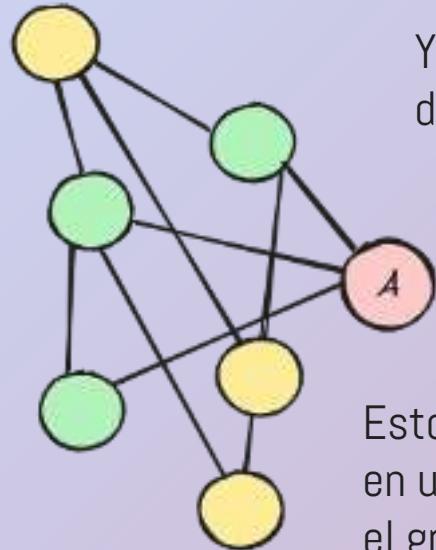
Y al igual que con la matriz de adyacencia, contamos al propio nodo como vecino, por eso debemos de sumarle la identidad, cada nodo tiene un vecino más (él mismo):

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

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

# Fundamento matemático en GCNs

GCN estandar



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

$$\begin{aligned} & \tilde{D}^{-1} \tilde{A} X W^T \\ & \tilde{A} \tilde{D}^{-1} X W^T \end{aligned}$$

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

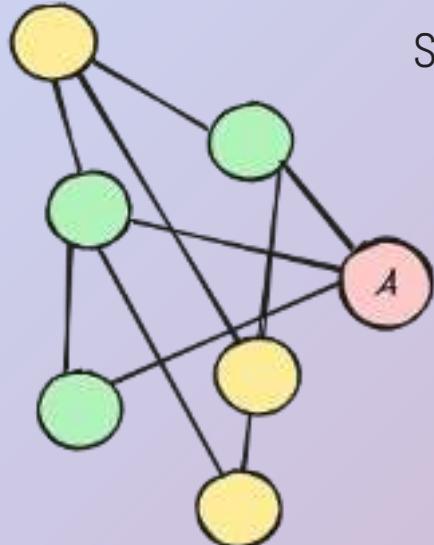
$$H = \tilde{D}^{-\frac{1}{2}} \tilde{A}^T \tilde{D}^{-\frac{1}{2}} X W^T$$

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.

# 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 código

# Práctica sobre GCNs

---

Colab Notebook:

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





# Caso concreto

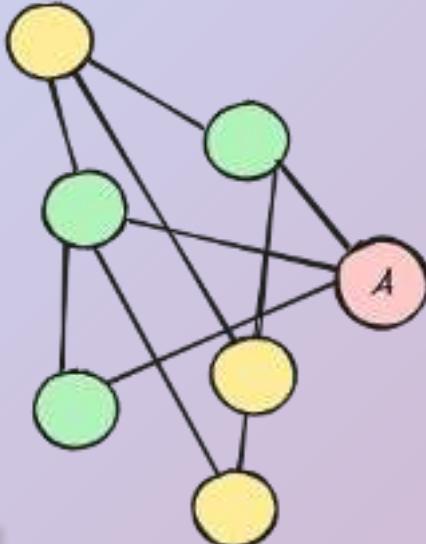
## Redes de Grafos con Atención (GNNs)

¿Cómo implementar los mecanismos de atención en las GNNs?

# 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.



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

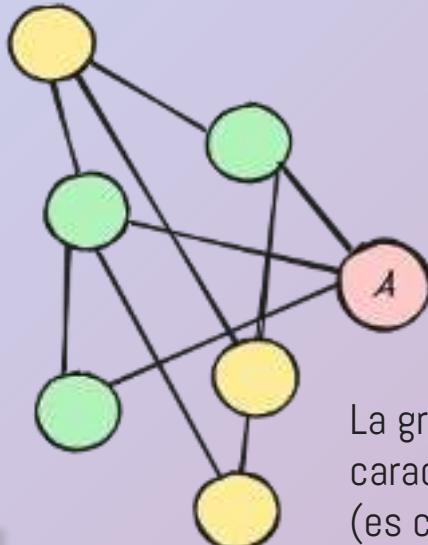
Pero esto no es cierto en general. A veces unos vecinos serán más importantes que otros. De hecho, se da mayor importancia a los nodos con menos vecinos gracias a la normalización usada:

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

# 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".



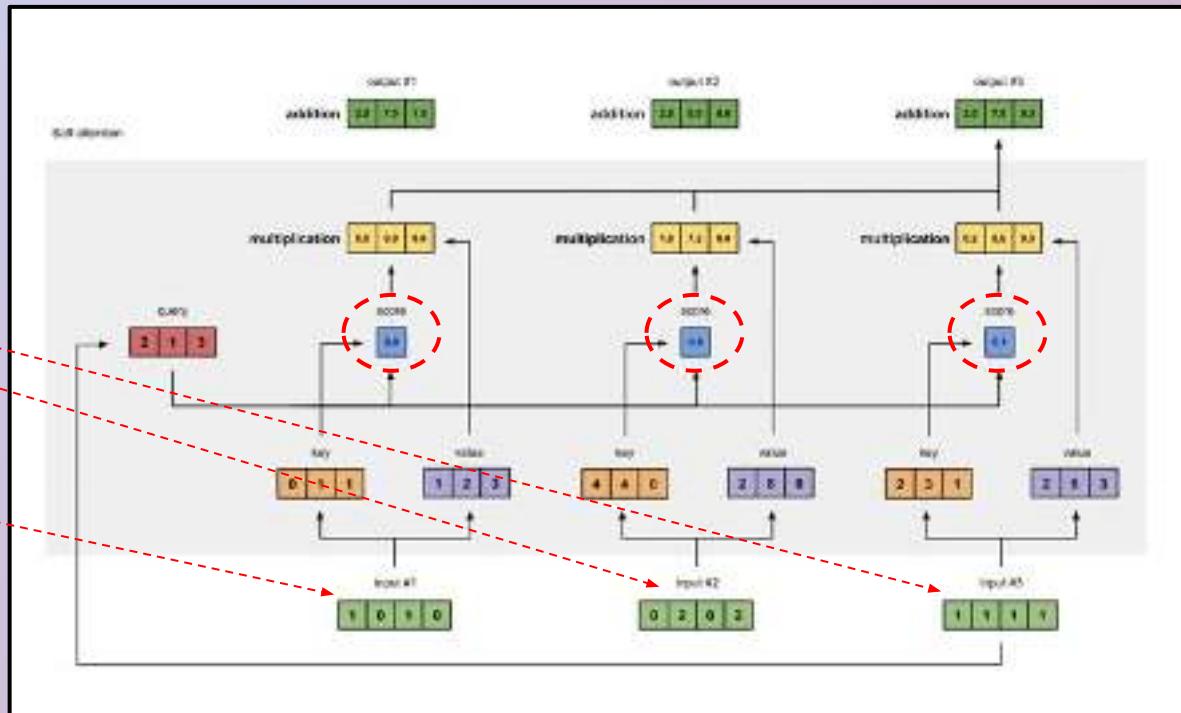
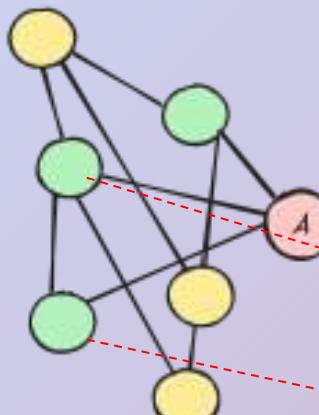
$$h_A = \sum_{i \in \mathcal{N}_A} \alpha_{iA} x_i W^T$$

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

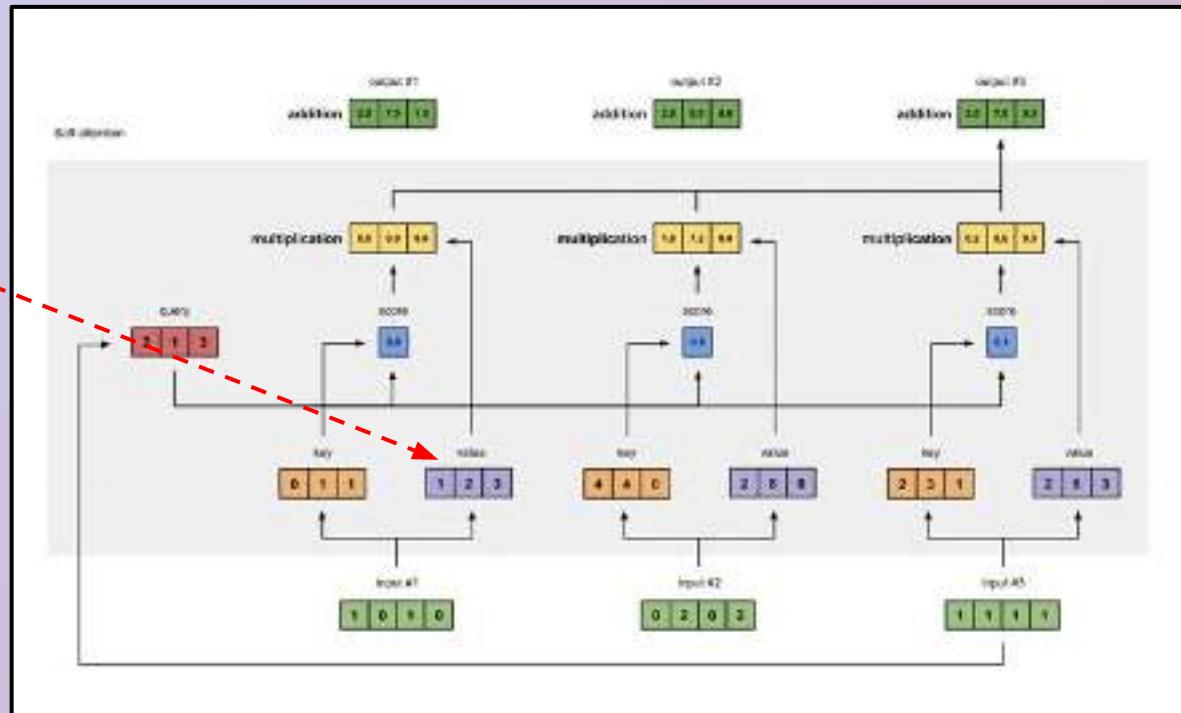
Breve inciso sobre el mecanismo de *self-attention* :



# Fundamento matemático en GANs

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

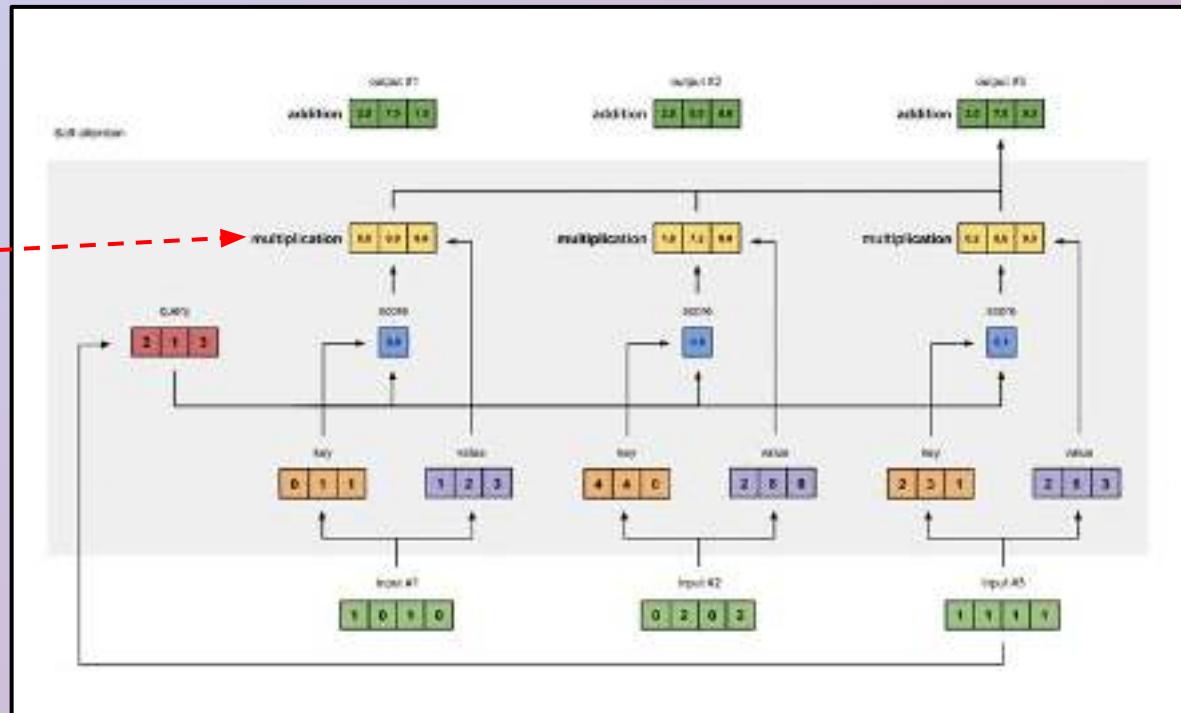
$$h_A = \sum_{i \in \mathcal{N}_A} \alpha_{iA} x_i W^T$$



# Fundamento matemático en GANs

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

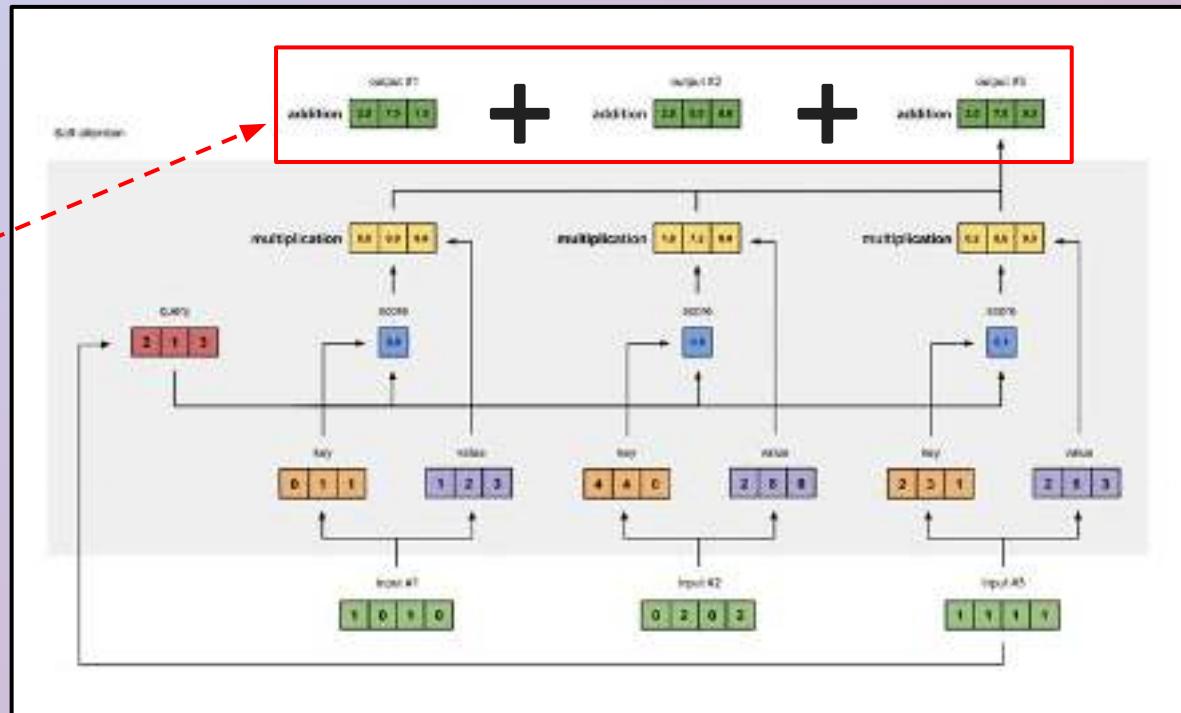
$$h_A = \sum_{i \in \mathcal{N}_A} \alpha_{iA} x_i W^T$$



# Fundamento matemático en GANs

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

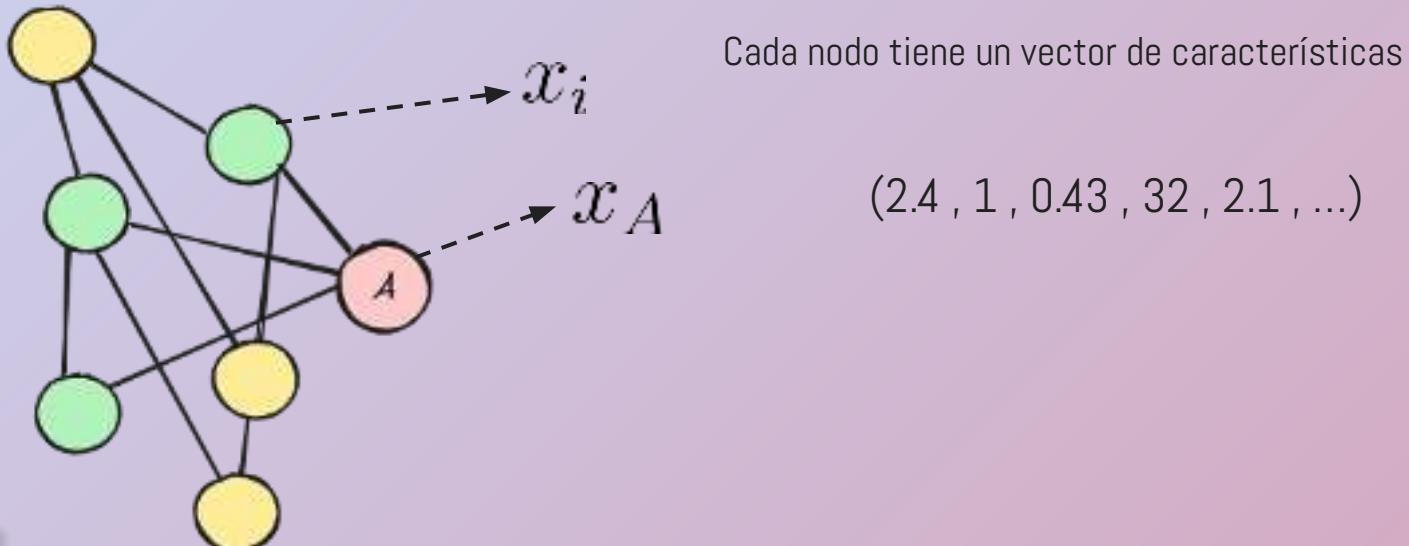
$$h_A = \sum_{i \in \mathcal{N}_A} \alpha_{iA} x_i W^T$$



# Fundamento matemático en GANs

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

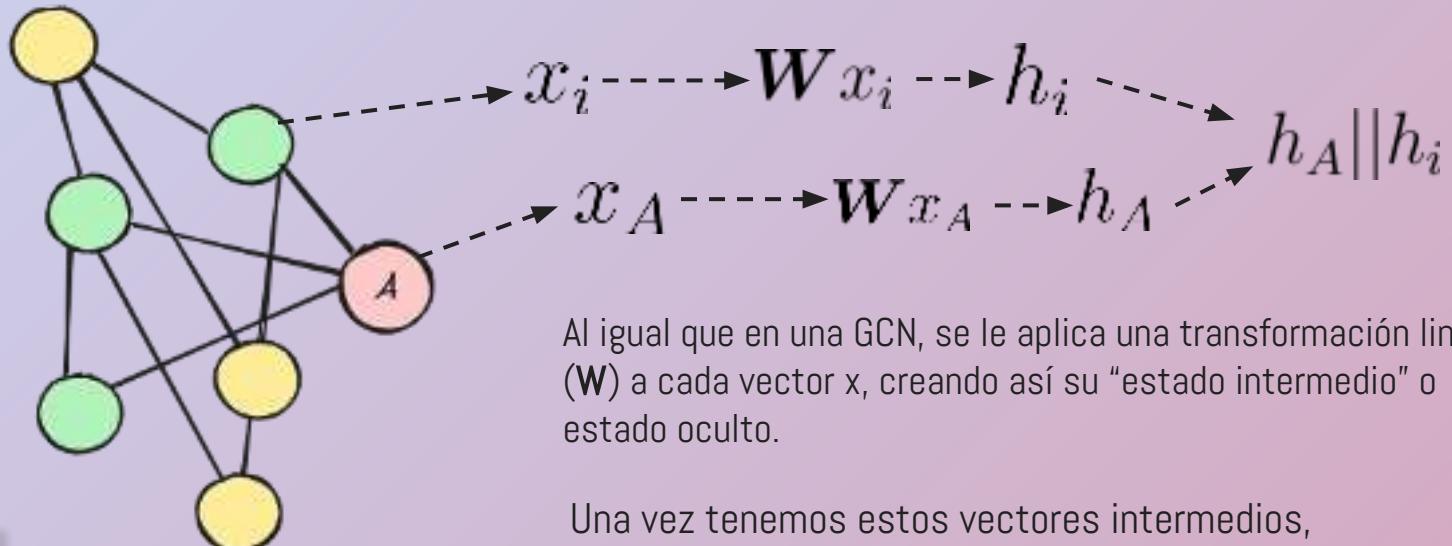
Sino que las ***attention scores*** se calcularán de la siguiente manera:



# 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:



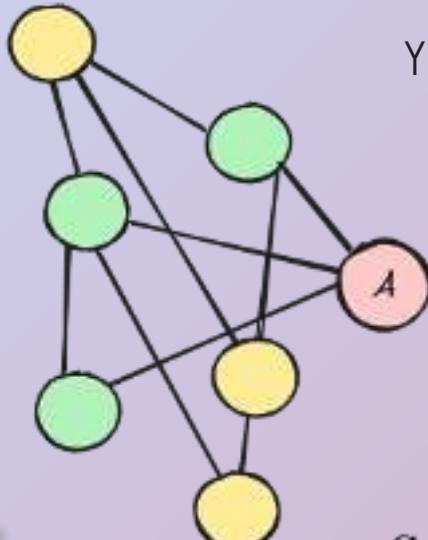
Al igual que en una GCN, se le aplica una transformación lineal ( $\mathbf{W}$ ) a cada vector  $x$ , creando así su “estado intermedio” o estado oculto.

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"

$$h_A \parallel h_i$$

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

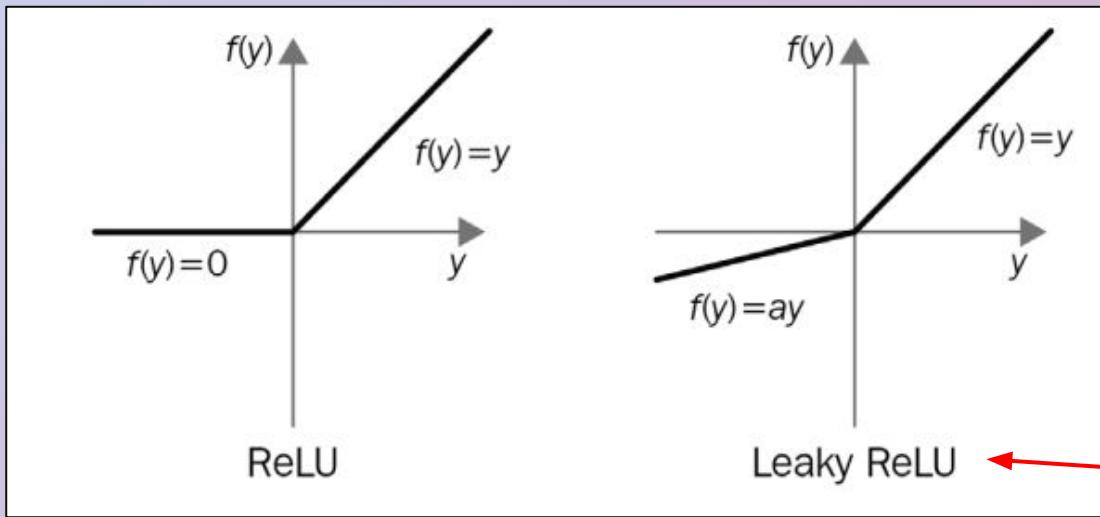
$$a_{iA} = W_{att}^T [h_A \parallel h_i] = W_{att}^T [\mathbf{W}x_A \parallel \mathbf{W}x_i]$$

*"Query"                            "Key"*

Es "a" no alpha!

# Fundamento matemático en GANs

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



$$e_{iA} = \text{LeakyReLU}(a_{iA})$$

# 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:

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

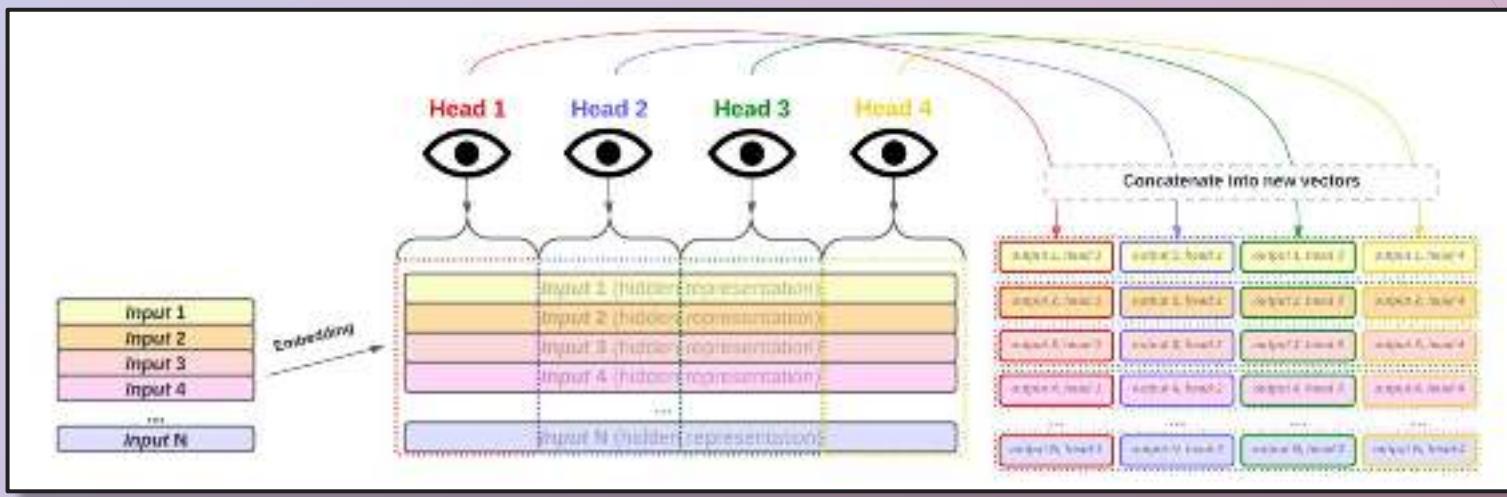
Ahora ya tenemos una medida de la importancia entre el nodo A y el nodo vecino i.

Recordemos que esto se calcula automáticamente, gracias a que entrenamos a la red a aprender  $W_{\text{att}}$ , y  $W$ , que hacen posible asignar “*attention scores*” entre vectores de características.

# 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.



# 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 .

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

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

# Fundamento matemático en GANs

---

Por último, el cálculo vectorial que realizamos es así:

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

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-9Nk7Bllalmmbl7MC?usp=sharing>



A complex network graph is visible in the background, composed of numerous small, semi-transparent nodes and edges in various colors like purple, green, and yellow, creating a sense of connectivity and data flow.

# Técnicas avanzadas

---

Cosas un poquito más complejas sobre las redes de grafos



# *GraphSAGE*

---

Escalabilidad en GNNs

# ¿Qué es GraphSAGE?

(Graph Sample and Aggregate)

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)

# ¿Qué es GraphSAGE?

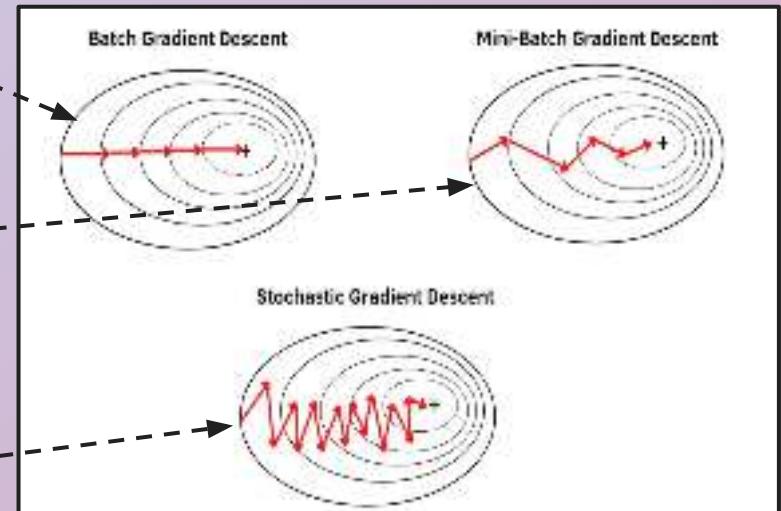
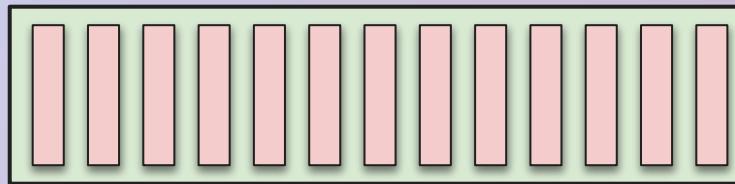
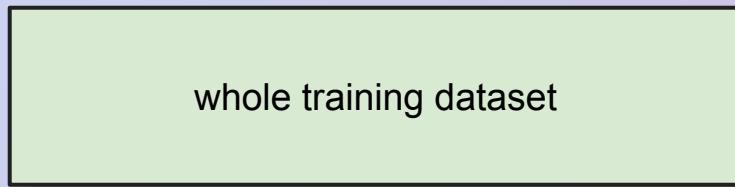
---

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

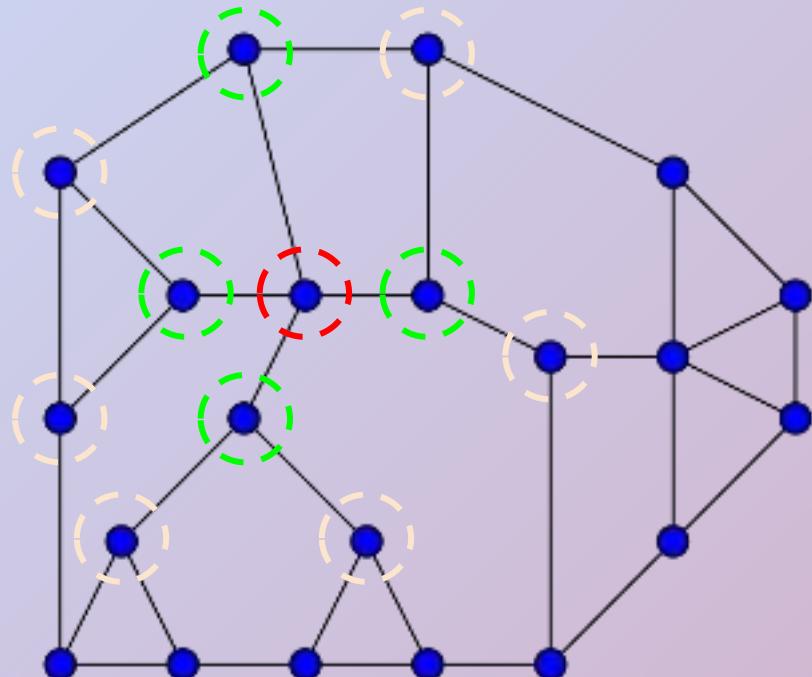
Para resolver este problema, nos preguntamos dos cosas:

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

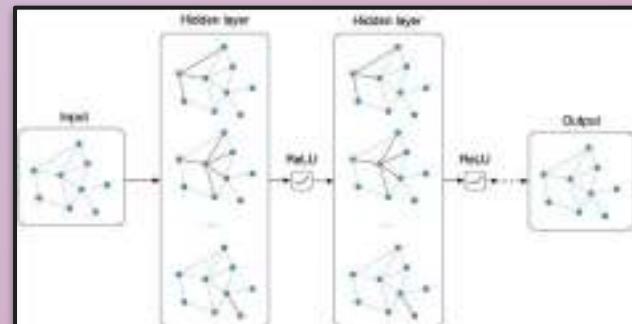
# Mini-batching



# Mini-batching en GNNs



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



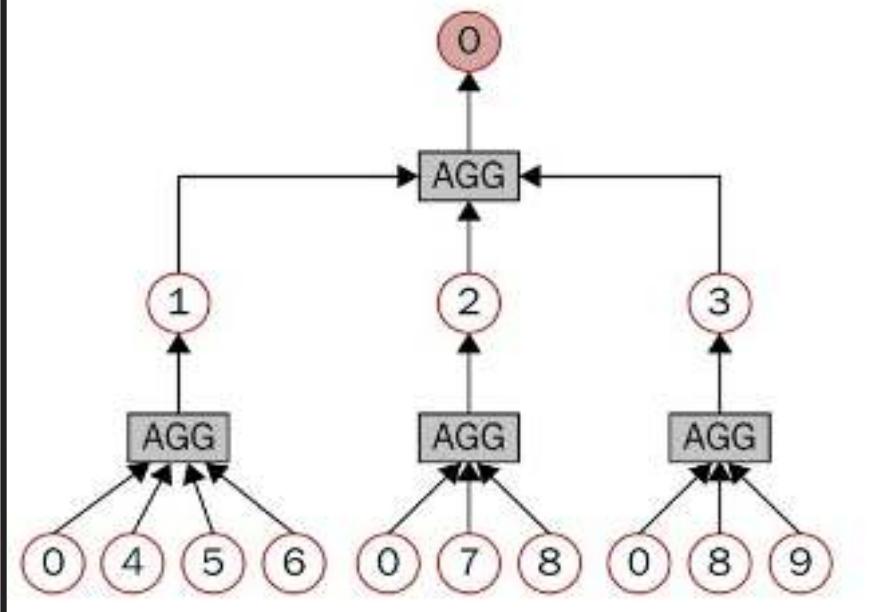
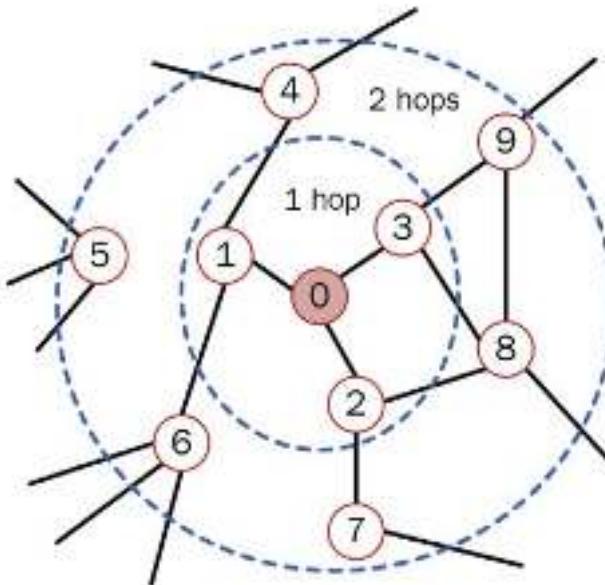
# Mini-batching en GNNs

---

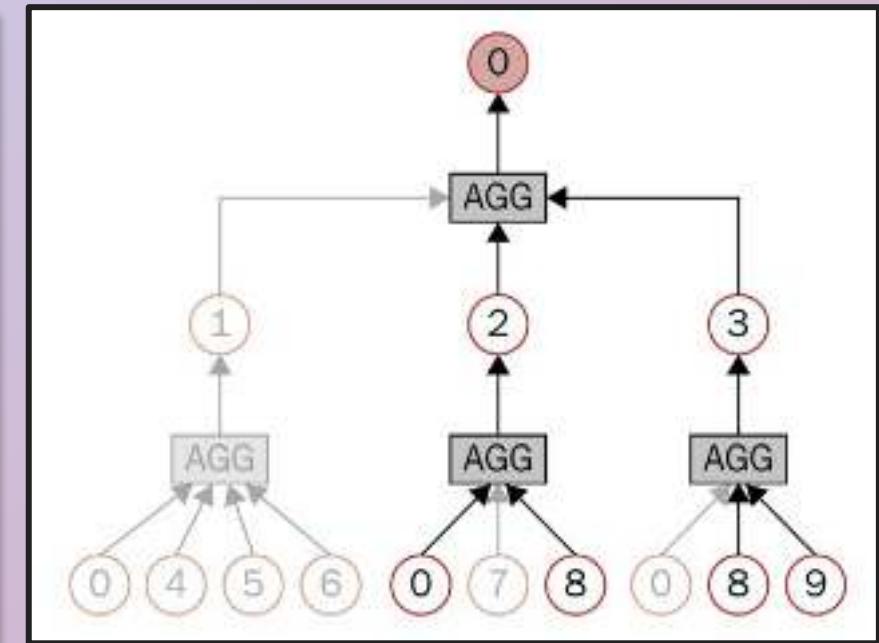
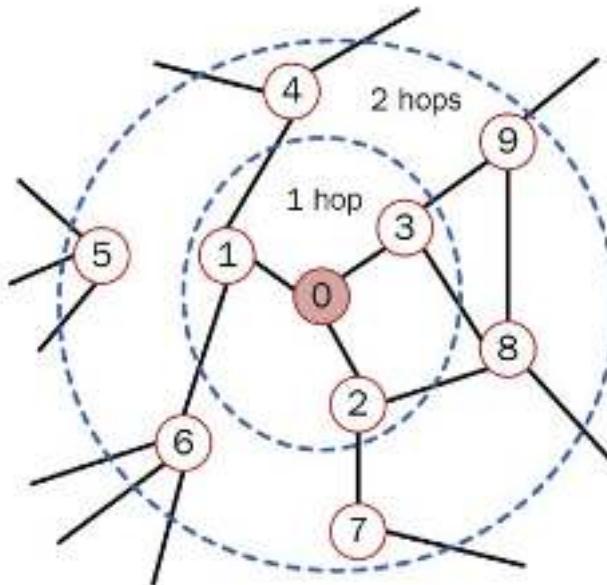
Y hay otro problema, si hubiera **nodos con muchísimas conexiones** (como una celebridad en una red social), se saturaría la estrategia de mini-batching. A estos nodos con una cantidad anormal de conexiones se les conoce como **hub-nodes**.

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



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

# 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.*

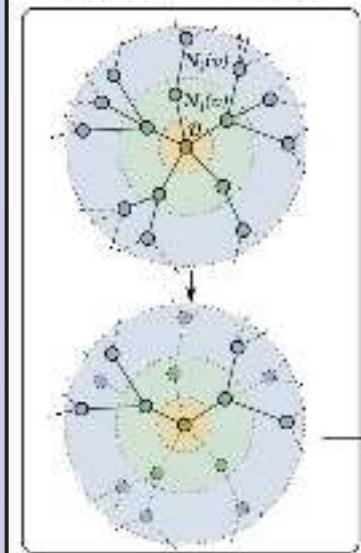
$$h_i' = \sigma (\mathbf{W} \cdot \text{mean}_{j \in \mathcal{N}_i}(h_j))$$

$$h_i' = \sigma (\mathbf{W}_1 h_i + \mathbf{W}_2 \cdot \text{mean}_{j \in \mathcal{N}_i}(h_j))$$

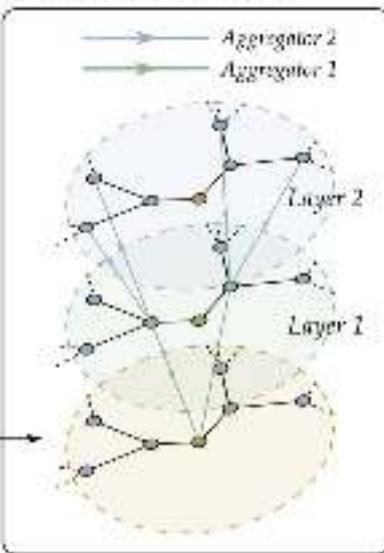
Para más información vease el artículo original:  
<https://cs.stanford.edu/~jure/pubs/graphsage-nips17.pdf>

# GraphSAGE overview

Neighbourhood sampling



Information aggregation



## Message Passing

### 1. Generación del mensaje

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

### 2. Agregación del mensaje

Todos los nodos reciben mensajes de sus vecinos mediante una función que no cuenta con un permutoequivalente, típicamente usando estos mensajes.

### 3. Actualización del estado

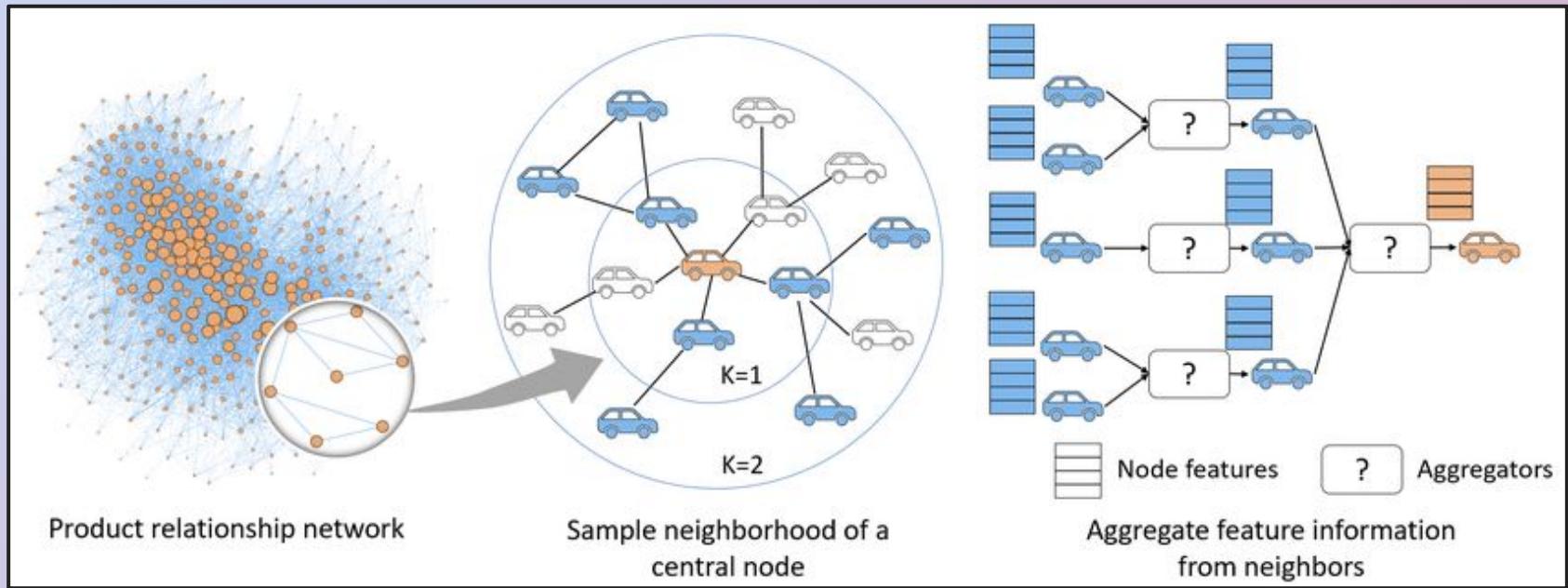
Actualiza sus propias características integrando estos mensajes agregados. Trabajando en bucles, el algoritmo para cada nodo actualiza sus estados de los mensajes.

**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 véase el artículo original:  
<https://cs.stanford.edu/~jure/pubs/graphsage-nips17.pdf>

# Prácticas

---

Veamos su aplicación en código

# Práctica sobre GraphSAGE

---

Colab Notebook:

[https://colab.research.google.com/drive/1qbQSTlcqkqNXG8T99Mve\\_Ek7BKI2Gnm?usp=sharing](https://colab.research.google.com/drive/1qbQSTlcqkqNXG8T99Mve_Ek7BKI2Gnm?usp=sharing)





# *Más aplicaciones prácticas*

---

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

# Prácticas

---

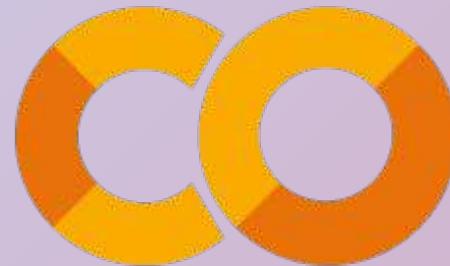
Veamos su aplicación en código

# Práctica sobre Aplicaciones GNNs

---

Colab Notebook:

[https://colab.research.google.com/drive/1r31pG3cKmxCeSw5NfnZWQT0CAAv\\_Q2k?usp=sharing](https://colab.research.google.com/drive/1r31pG3cKmxCeSw5NfnZWQT0CAAv_Q2k?usp=sharing)



# Gracias por vuestra atención

---

¿Preguntas ?



LinkedIn: [Arturo Sirvent Fresneda](#)