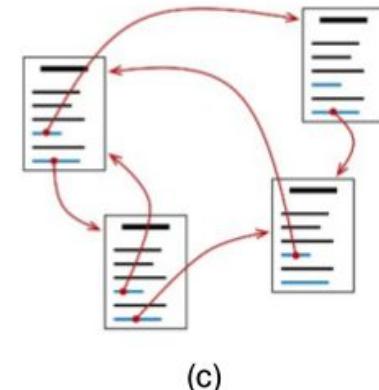
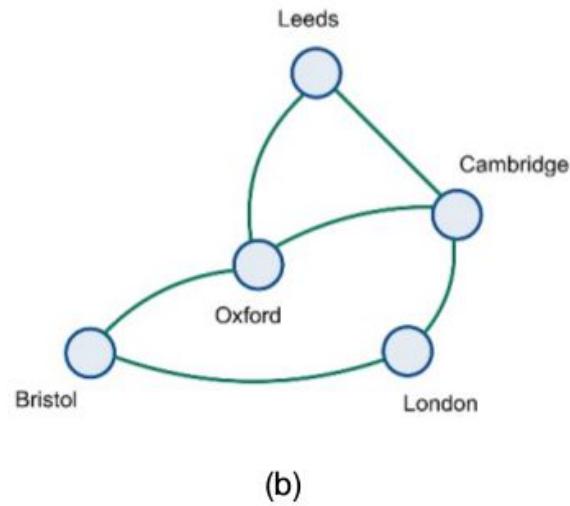
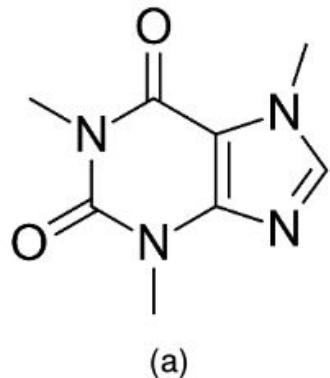


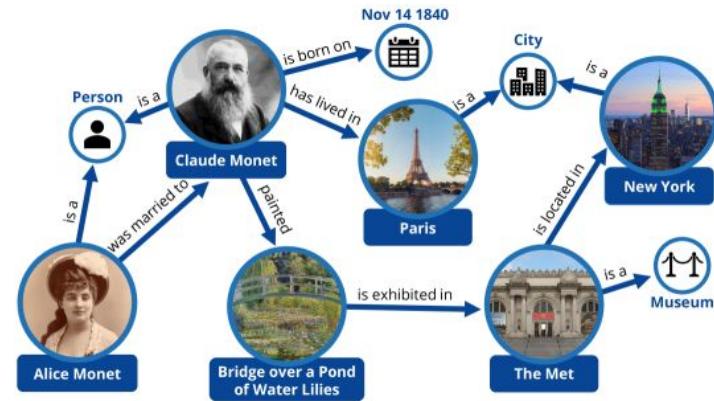
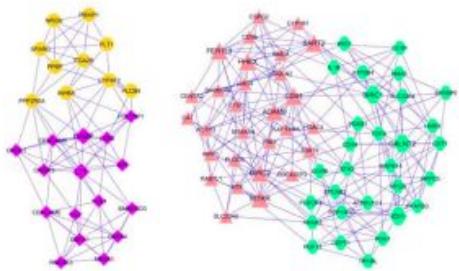
Redes Neuronales basadas en Grafos

Módulo 7 / Clase 4 - Teoria

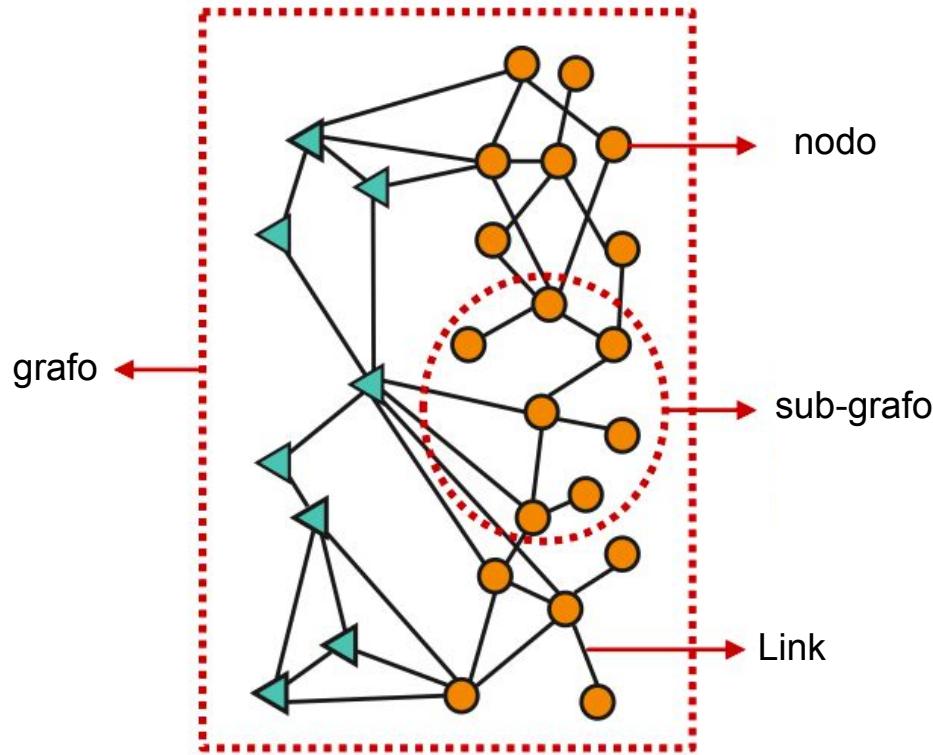
Ejemplos de datos con estructura de Grafo



Ejemplos de datos con estructura de Grafo



Tipo de predicción (Task)



Tipos de predicciones sobre grafos:

Pueden realizarse predicciones a nivel de:

- Nodo (ej. clasificación de documentos)
- Arista (ej. tarea de completado de links)
- Sub-grafo (ej. encontrar comunidades en el grafo)
- Grafo (ej. predicción si una molécula es soluble en agua o no)



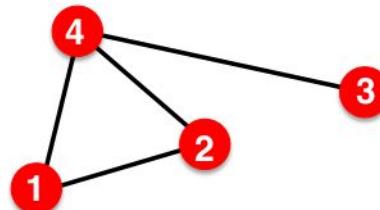
Grafo

Estructura matemática y conceptual que se utiliza para representar relaciones entre diferentes elementos

$$\mathcal{G}(\mathcal{V}, \mathcal{E})$$

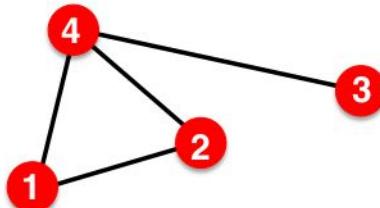
\mathcal{V} vértices o nodos

\mathcal{E} aristas o links



Representación de grafo: Matriz de adyacencia

grafo no dirigido



$A_{ij} = 1$ si hay una conexión entre i y j

$A_{ij} = 0$ si no hay conexión

$$k_i = \sum_{j=1}^N A_{ij}$$

$$k_j = \sum_{i=1}^N A_{ij}$$

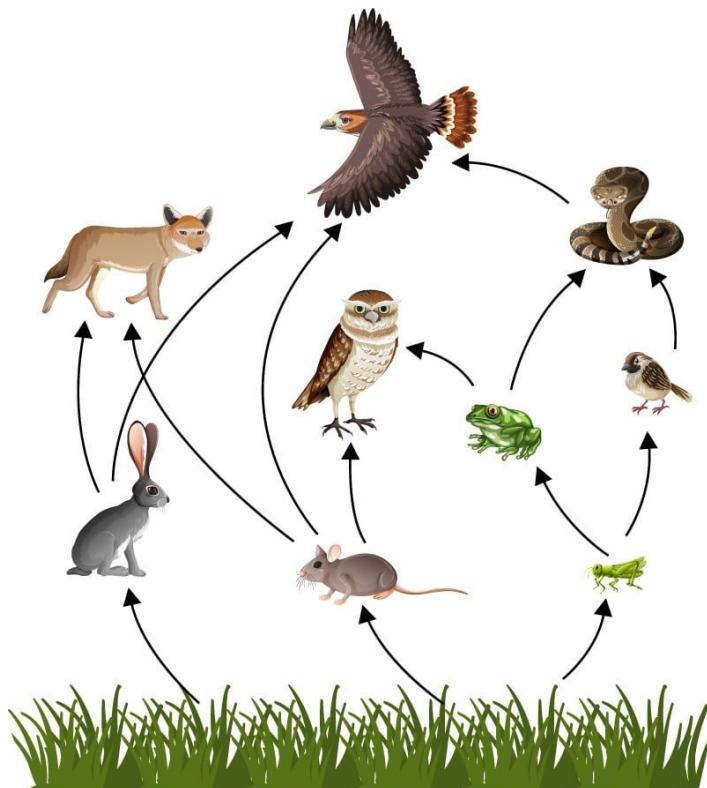
$$L = \frac{1}{2} \sum_{i=1}^N k_i = \frac{1}{2} \sum_{ij} A_{ij}$$

$$A_{ij} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

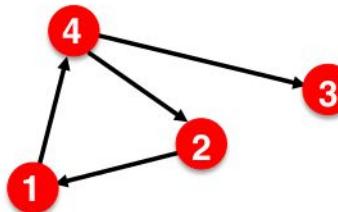
$A_{ij} = A_{ji}$
matriz simétrica



Representación de grafo: Matriz de adyacencia



grafo dirigido



$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

$$A_{::} \neq A_{::}$$

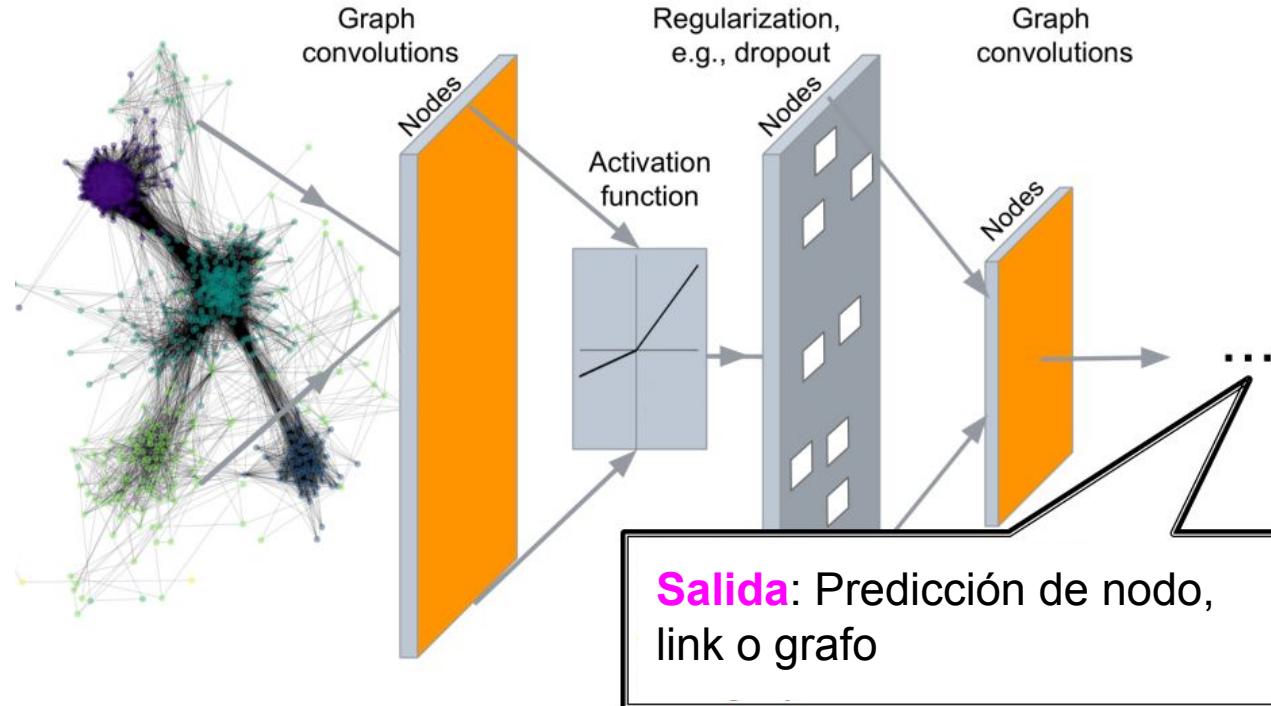
matriz no simétrica

$$k_i^{out} = \sum_{j=1}^N A_{ij}$$

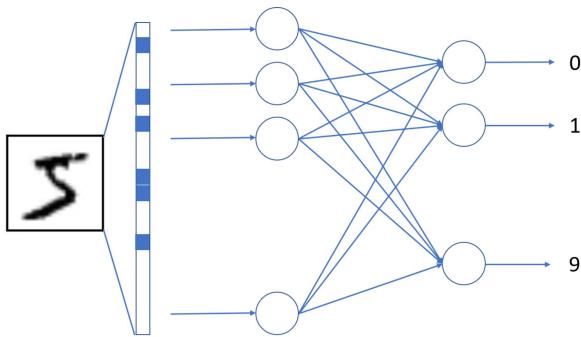
$$k_j^{in} = \sum_{i=1}^N A_{ij}$$

$$L = \sum_{i=1}^N k_i^{in} = \sum_{j=1}^N k_j^{out} = \sum_{i,j} A_{ij}$$

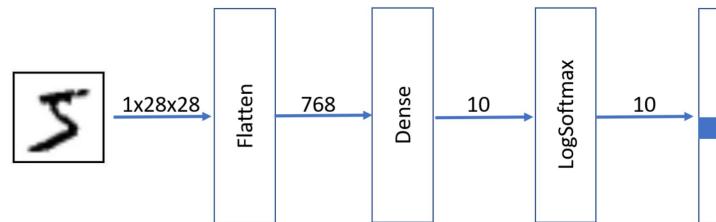
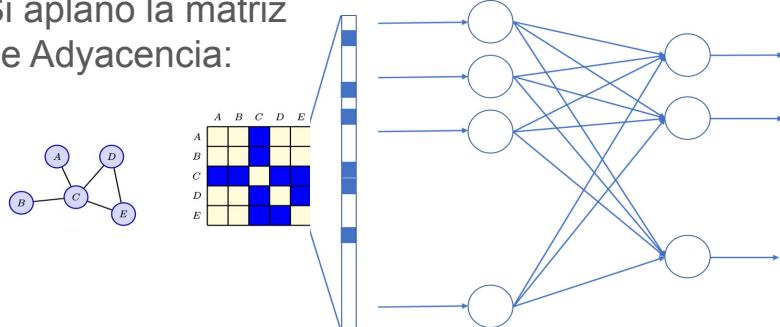
¿Cómo aplicar Deep Learning a datos con estructura de grafo?



¿Puedo pasar el grafo por una Red Neuronal completamente conectada ?



Si aplano la matriz de Adyacencia:



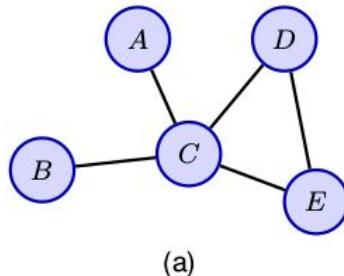
Problemas:

- No aplicable a grafos de diferente tamaño
- Sensible al orden de los nodos en el grafo
- Demasiados parámetros

¿Cómo adaptar las arquitecturas disponibles para grafos?

- Entradas de longitud variable
- Invarianza y Equivarianza a las permutaciones
- Escalabilidad

Uso parámetros compartidos



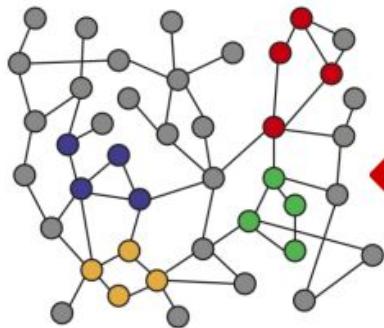
(b)

	A	B	C	D	E
A	Y	Y			
B		Y			
C			Y	Y	Y
D				Y	Y
E					Y

(c)

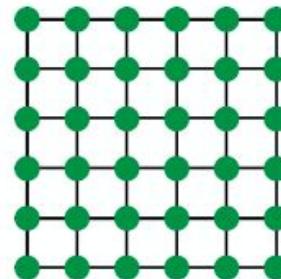
	C	E	A	D	B
C	Y				
E		Y			
A			Y		
D				Y	
B					Y

Imagen es un caso especial de Grafo



Grafo

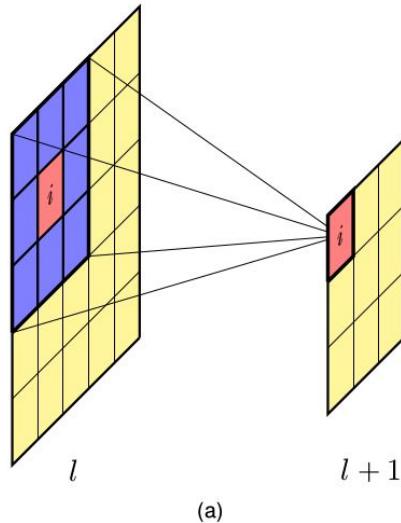
Tamaño arbitrario
Estructura topológica compleja



Imagen

Tamaño definido
Estructura espacial fija y bien definida

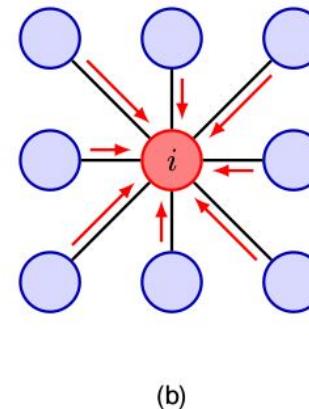
¿Podemos aprovechar la Idea de Convolución?



$$z_i^{(l+1)} = f \left(\sum_j w_j z_j^{(l)} + b \right)$$

NO equivariante

pasaje de mensaje



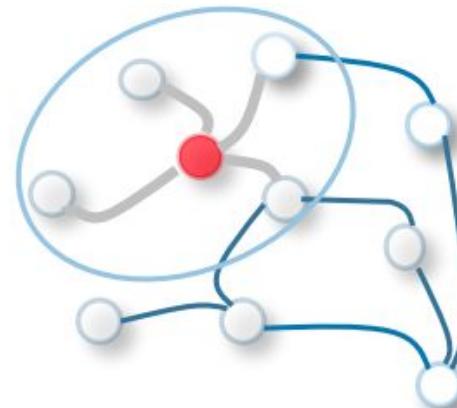
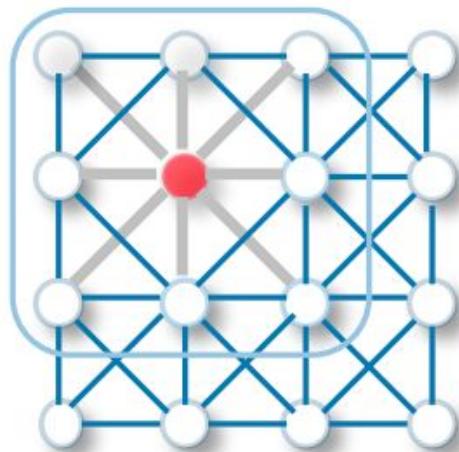
(b)

parámetros
compartidos

$$z_i^{(l+1)} = f \left(w_{\text{neigh}} \sum_{j \in \mathcal{N}(i)} z_j^{(l)} + w_{\text{self}} z_i^{(l)} + b \right)$$

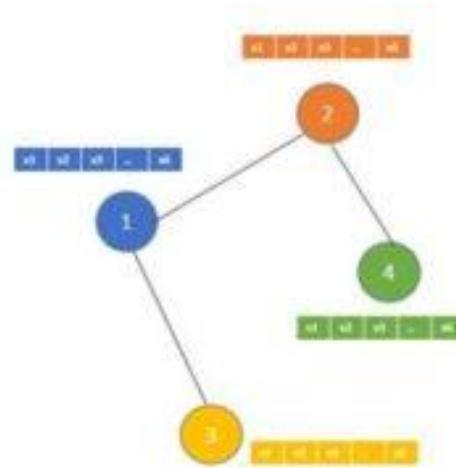
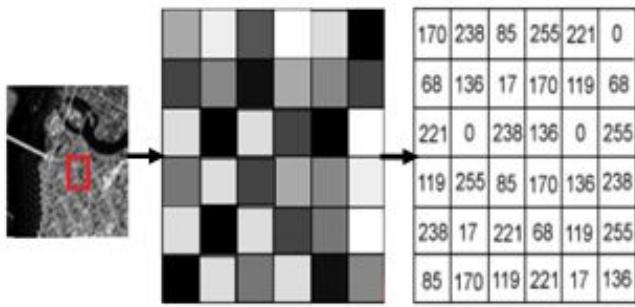
Equivariante

De una imagen a un grafo: Topología del grafo



$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Características asociadas a los nodos



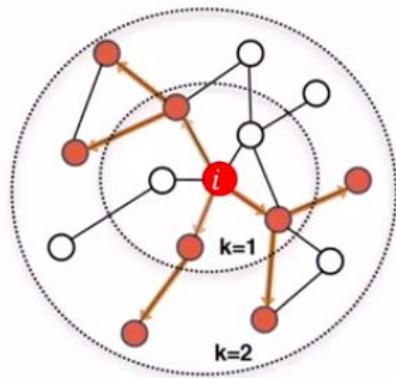
$$H^{(0)} = X$$

$h_1^0 = x_1$
 $h_2^0 = x_2$
 $h_3^0 = x_3$
 $h_4^0 = x_4$

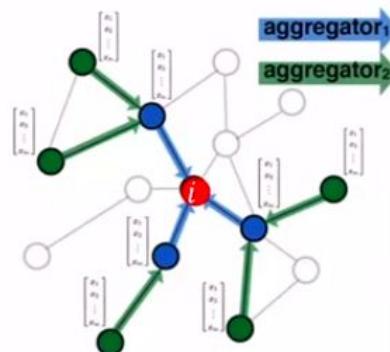
A 4x4 matrix labeled $H^{(0)} = X$. The matrix has four rows, each representing a feature vector h_i^0 for $i = 1, 2, 3, 4$. The rows are colored blue, orange, yellow, and green respectively, matching the colors of the feature vectors above the graph nodes.

Convolución en grafos

Podemos pensar en un proceso en dos etapas:

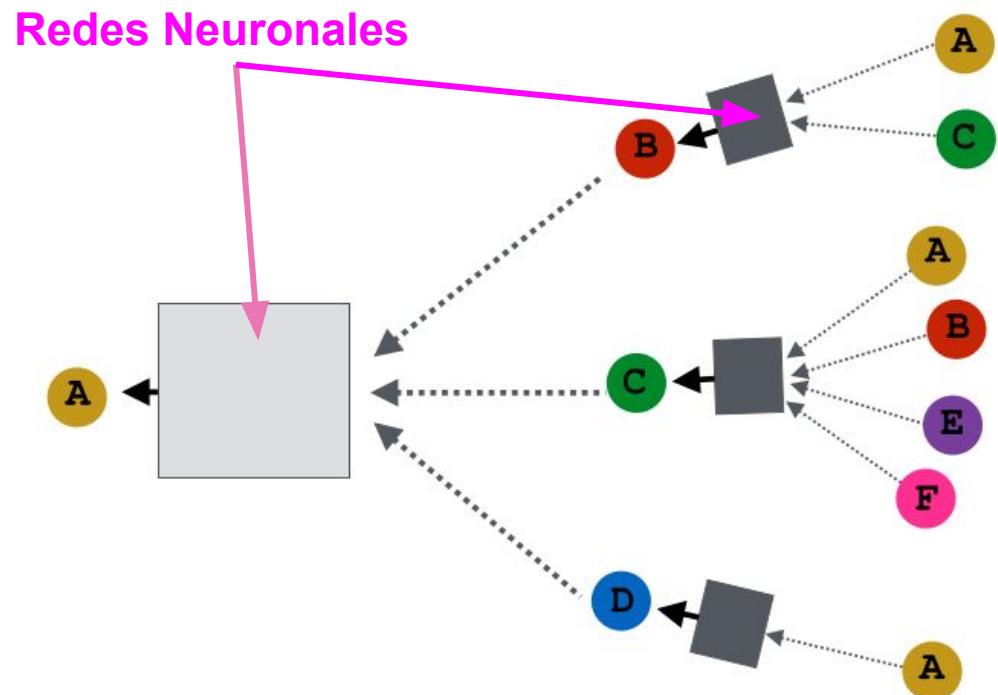
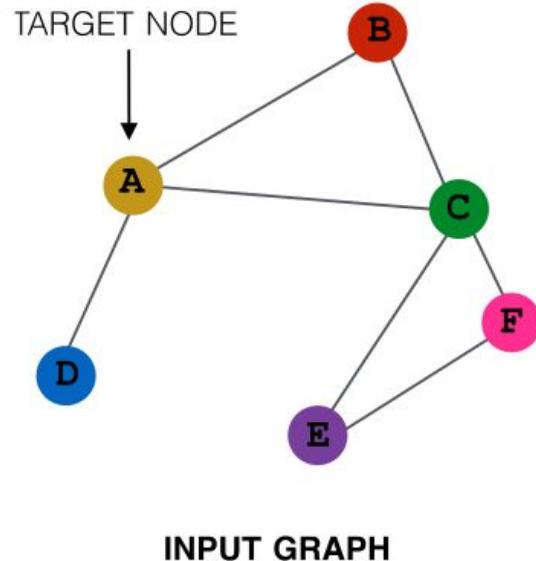


1º) Definir el grafo de computación de cada nodo (definido por su “vecindario”)



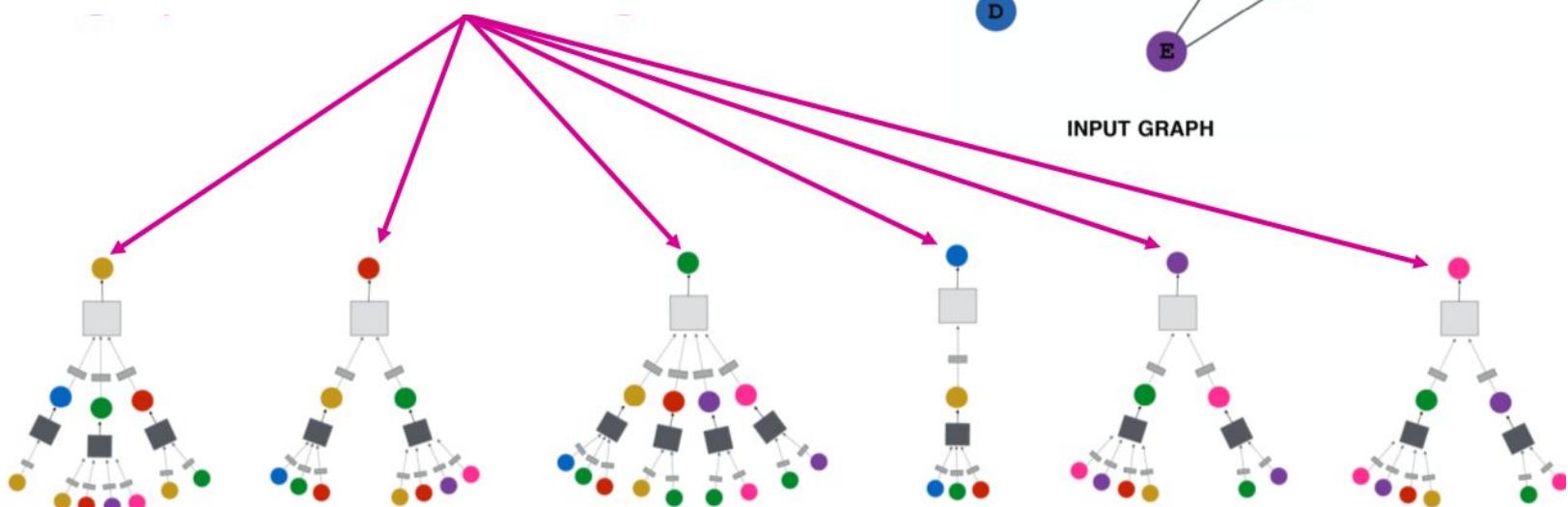
2º) Propagar la información a través del grafo (convolución)

1- Definir el grafo de computación



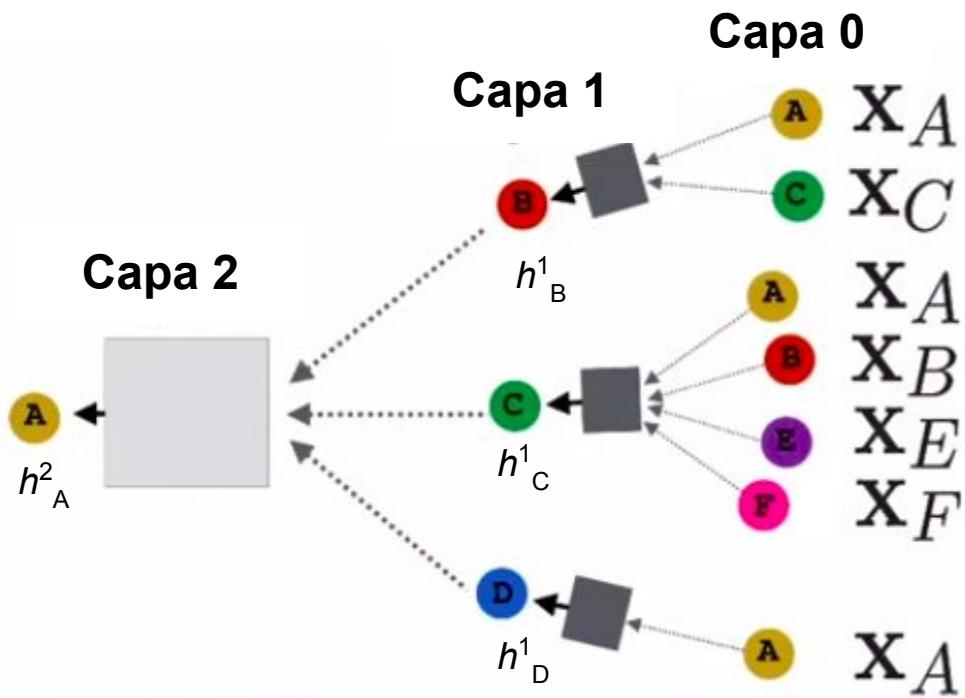
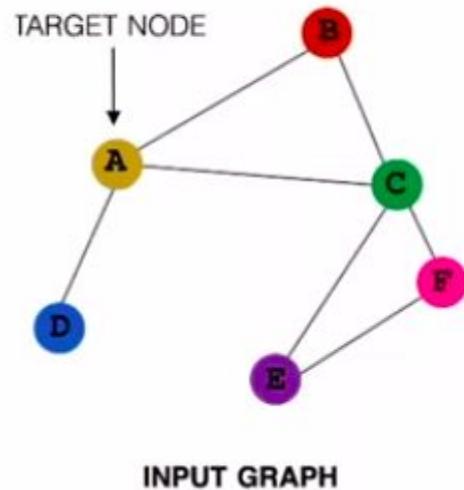
Grafo de computación del nodo A

Cada nodo define su propia red neuronal

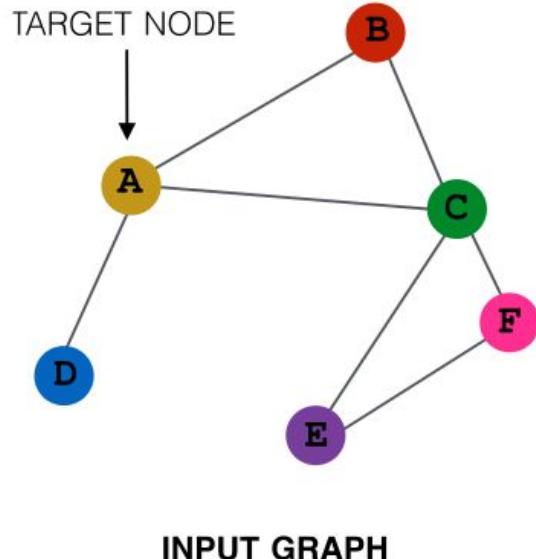


La GNN debe aprender simultáneamente los parámetros de cada una de estas arquitecturas !!

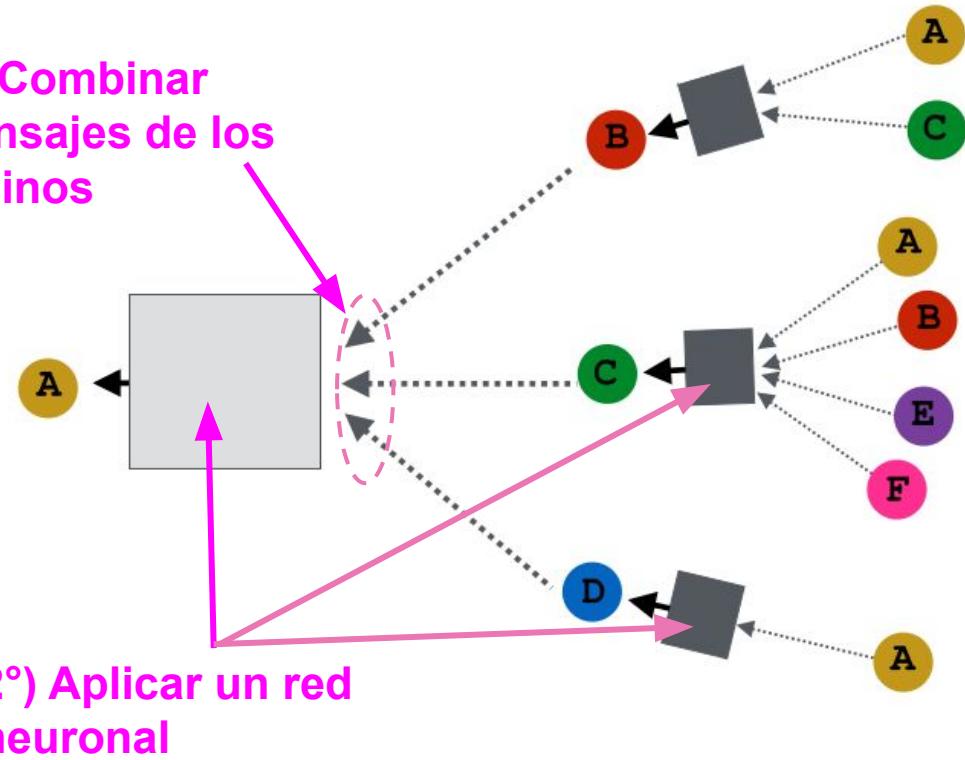
2-Propagar información de vecinos



2- Propagar información de vecinos



1º) Combinar
mensajes de los
vecinos



Actualización/Propagación de la información

$$h_v^0 = x_v$$

$$h_v^{(l+1)} = \sigma(W_l \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|} + B_l h_v^{(l)}) \quad \forall l \in \{0, \dots, L-1\}$$

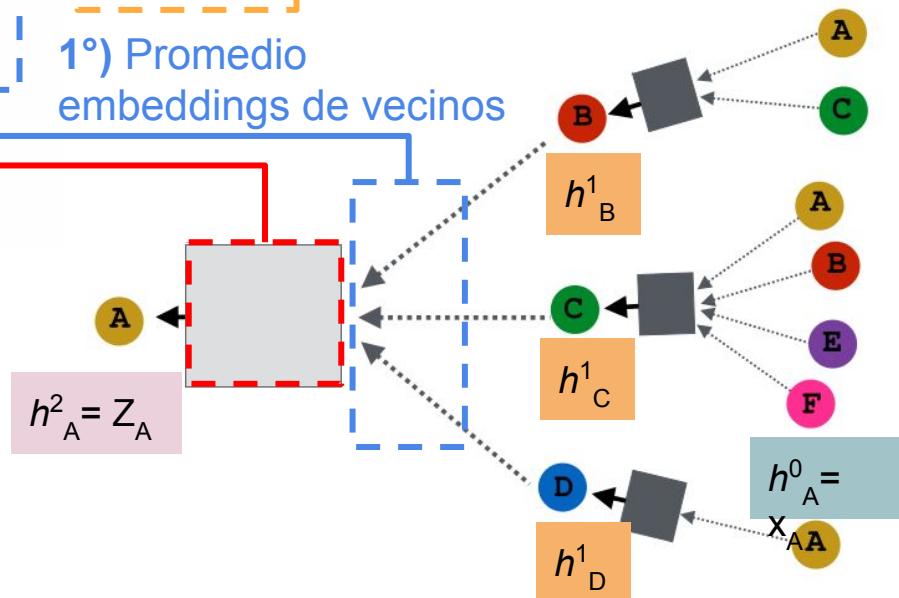
$$z_v = h_v^{(L)}$$

2º) Transformación no-lineal

Embedding del mismo nodo en la capa previa

1º) Promedio embeddings de vecinos

Promedio no depende del orden de los vecinos!!



Formulación matricial

La agregación de vecinos puede llevarse a cabo eficientemente mediante operaciones con matrices

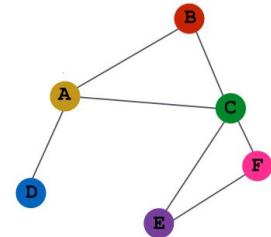
$$h_v^{(l+1)} = \sigma(W_l \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|} + B_l h_v^{(l)})$$

A

H⁽⁰⁾ = X

H⁽¹⁾

$$\begin{array}{c} \leftarrow N \text{ nodos} \rightarrow \\ \uparrow \downarrow N \text{ nodos} \\ \begin{matrix} \text{A} \\ \text{B} \\ \text{C} \\ \text{D} \\ \text{E} \\ \text{F} \end{matrix} \end{array} \cdot \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{array}{c} \leftarrow M \text{ features} \rightarrow \\ \uparrow \downarrow N \text{ nodos} \\ \begin{matrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{matrix} \end{array} = \begin{pmatrix} 2 & 1 \\ 2 & 1 \\ 2 & 2 \\ 1 & 0 \\ 0 & 2 \\ 0 & 2 \end{pmatrix}$$



Problemas con la formulación matricial previa:

- Gradientes que desaparecen (vanishing gradients) o que explotan (exploding gradients). Nodos con pocas/muchas conexiones tendrán valores pequeño/grandes en su representación de características.

$$\sum_{u \in N(v)} h_u^{(l)} = A_v H^{(l)}$$

- La representación agregada de un nodo no incluye sus propias características

$$h_v^{(l+1)} = \sigma(W_l \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|} + B_l h_v^{(l)})$$

A

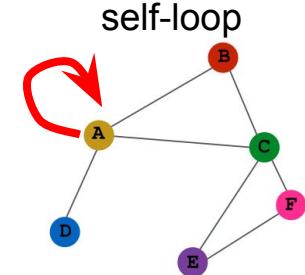
0	1	1	1	0	0
1	0	1	0	0	0
1	1	0	0	1	1
1	0	0	0	0	0
0	0	1	0	0	1
0	0	1	0	1	0

Self-loops

$$h_v^{(l+1)} = \sigma(W_l \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|} + B_l h_v^{(l)})$$

- La representación agregada de un nodo no incluye sus propias características
- Solución: añadir una conexión o bucle propio (self-loop) para que el nodo también considere sus propias características en la actualización.
- En la práctica, esto puede hacerse sumando la matriz identidad I a la matriz de adyacencia A antes de actualizar la representación de los nodos.

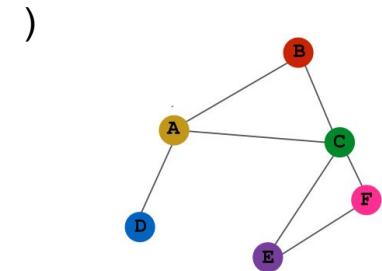
$$\begin{matrix} A \\ \boxed{\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}} \end{matrix} + \begin{matrix} I \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix} = \begin{matrix} \hat{A} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{matrix}$$



Promedio de los mensajes

Promediamos los mensajes para que su magnitud NO dependa del número de vecinos

$$h_v^{(l+1)} = \sigma(W_l \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|})$$



$$\begin{array}{c}
 D^{-1} \\
 \text{---} \\
 \begin{matrix}
 \textcolor{orange}{A} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\
 \textcolor{red}{B} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\
 \textcolor{green}{C} & 0 & 0 & \frac{1}{5} & 0 & 0 & 0 \\
 \textcolor{blue}{D} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\
 \textcolor{purple}{E} & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 \\
 \textcolor{pink}{F} & 0 & 0 & 1 & 0 & 0 & \frac{1}{3}
 \end{matrix}
 \cdot
 \begin{pmatrix}
 1 & 1 & 1 & 1 & 0 & 0 \\
 1 & 1 & 1 & 0 & 0 & 0 \\
 1 & 1 & 1 & 0 & 1 & 1 \\
 1 & 0 & 0 & 1 & 0 & 0 \\
 0 & 0 & 1 & 0 & 1 & 1 \\
 0 & 0 & 1 & 0 & 1 & 1
 \end{pmatrix}
 = \\
 \begin{pmatrix}
 \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\
 \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\
 \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 & \frac{1}{5} & \frac{1}{5} \\
 \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 \\
 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\
 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3}
 \end{pmatrix}
 \end{array}$$

$$D^{-1} = \frac{1}{|N(v)|}$$

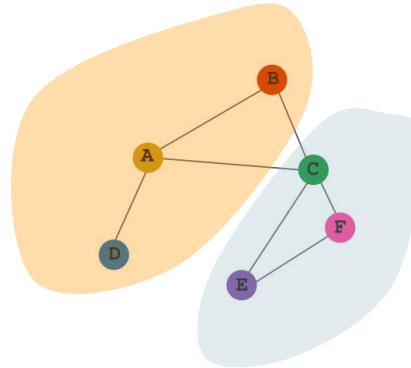
Toy example

$$D^{-1}\hat{A} \cdot \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{3}{4} & \frac{1}{4} \\ \frac{2}{3} & \frac{1}{3} \\ \frac{2}{5} & \frac{3}{5} \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

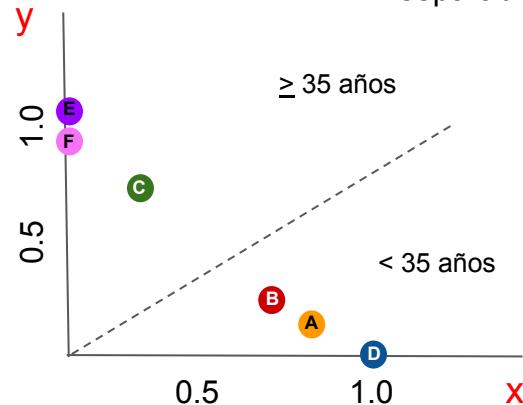
x y

Edad en codificación one-hot

v 35 **AI** 35



nodos linealmente separables en el espacio de embeddings

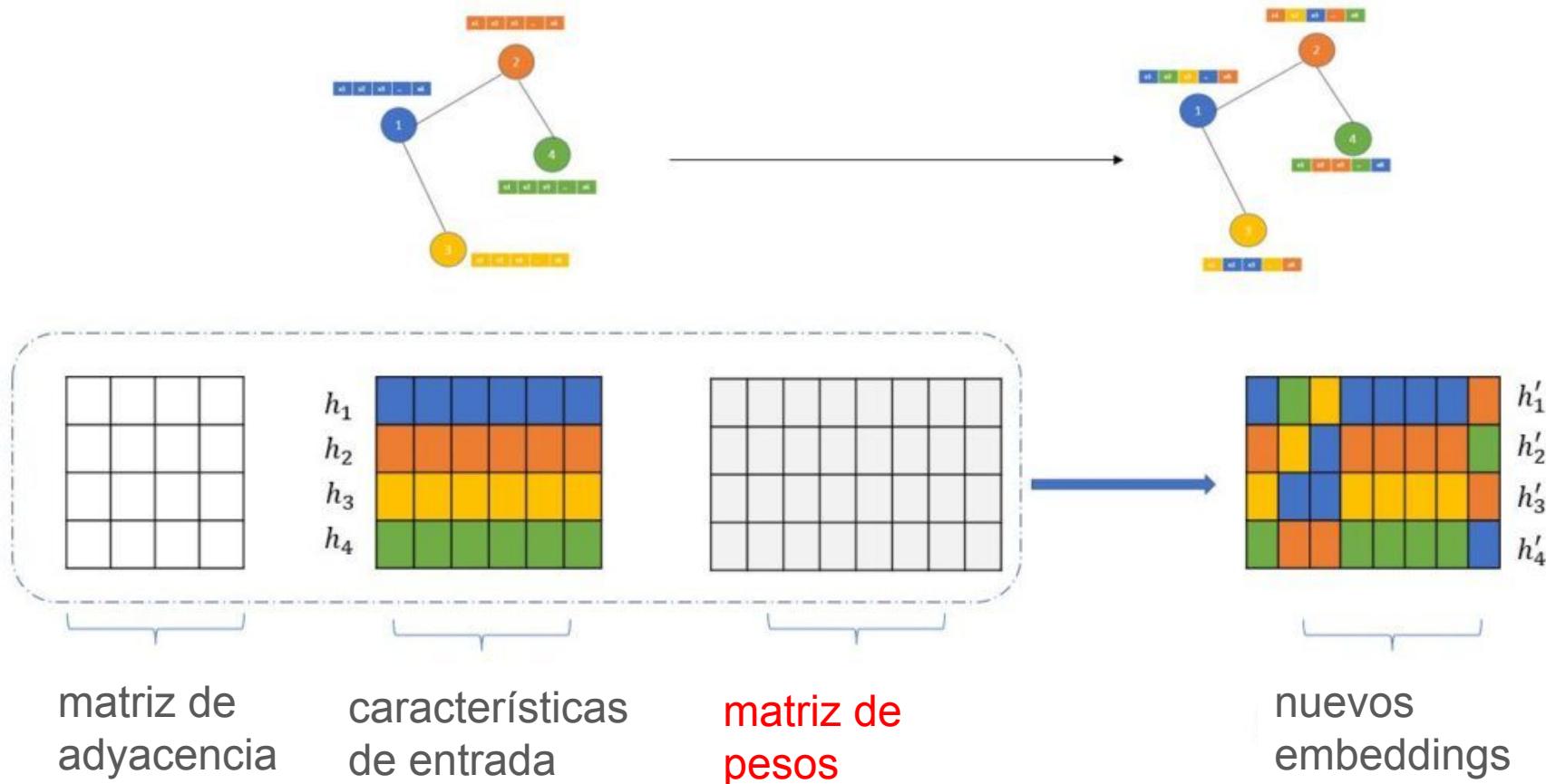


Los embeddings de la GCN representan correctamente la estructura y las relaciones del grafo, incluso antes de la fase de entrenamiento!!

aún no ajustamos los parámetros de la GCN

$$h_v^{(l+1)} = \sigma(W_l \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|})$$

Matriz de Pesos



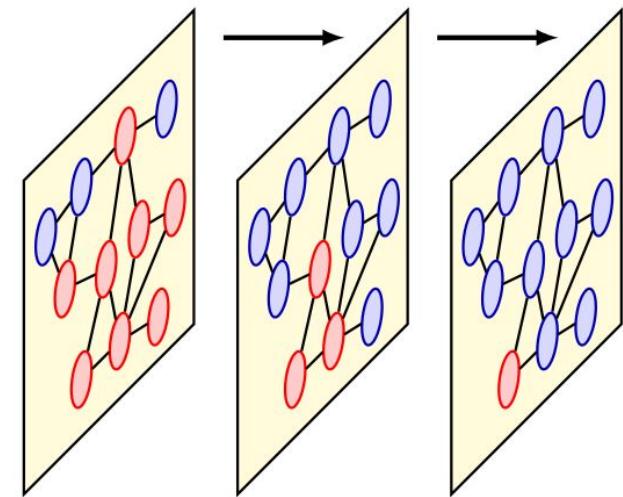
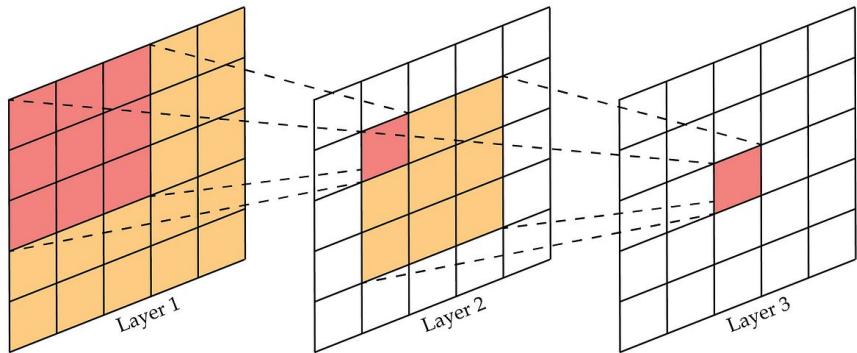
Campo receptivo

$$\mathbf{H}^{(1)} = \mathbf{F}(\mathbf{X}, \mathbf{A}, \mathbf{W}^{(1)})$$

$$\mathbf{H}^{(2)} = \mathbf{F}(\mathbf{H}^{(1)}, \mathbf{A}, \mathbf{W}^{(2)})$$

$$\vdots = \vdots$$

$$\mathbf{H}^{(L)} = \mathbf{F}(\mathbf{H}^{(L-1)}, \mathbf{A}, \mathbf{W}^{(L)})$$



Entrenamiento

Queremos minimizar la función de pérdida:

$$\min_{\theta} \mathcal{L}(y, f(z_v))$$

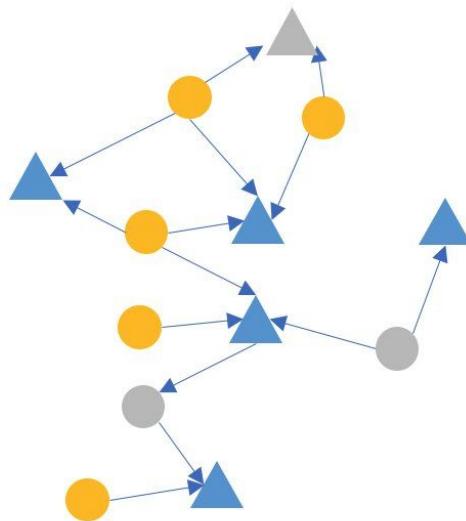
label del nodo embeddings de la última
capa convolucional

Por ejemplo para clasificación usamos Cross-entropy loss:

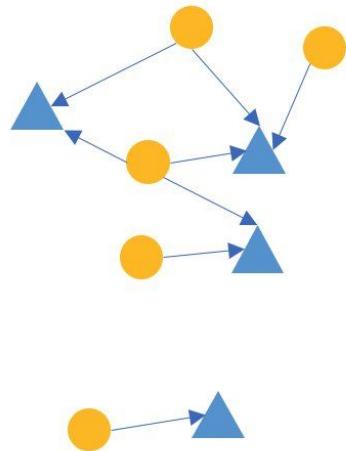
$$\mathcal{L} = \sum_{v \in V} y_v \log(\sigma(z_v^T \theta)) + (1 - y_v) \log(1 - \sigma(z_v^T \theta))$$

Aprendizaje Inductivo y Transductivo

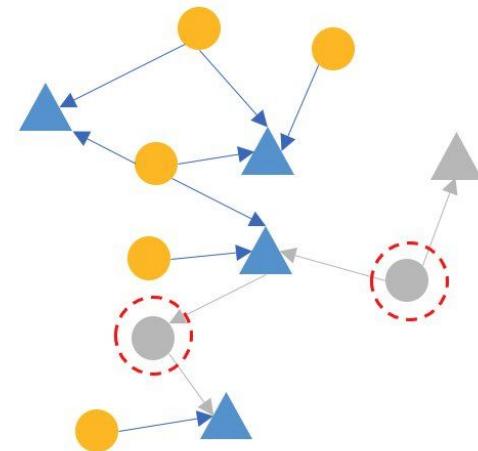
Inferencia durante el entrenamiento



Entrenamiento



Inferencia

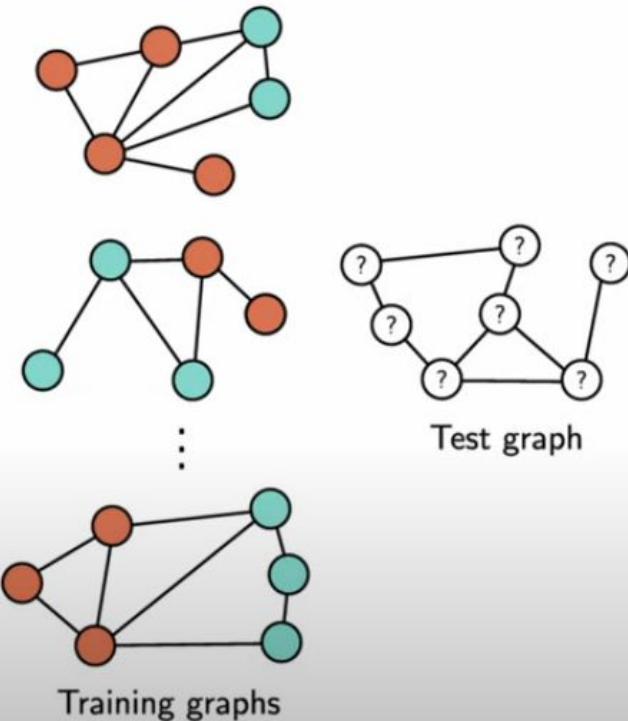


Modo Transductivo

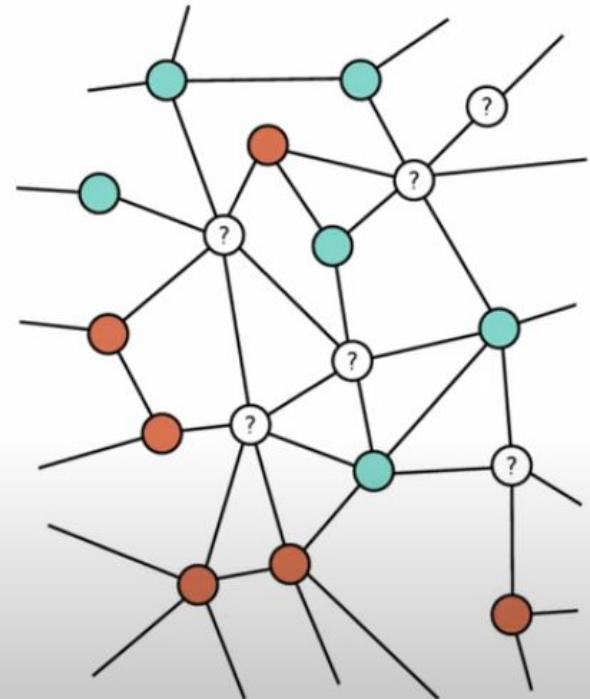
Modo Inductivo

Aprendizaje Inductivo versus Transductivo

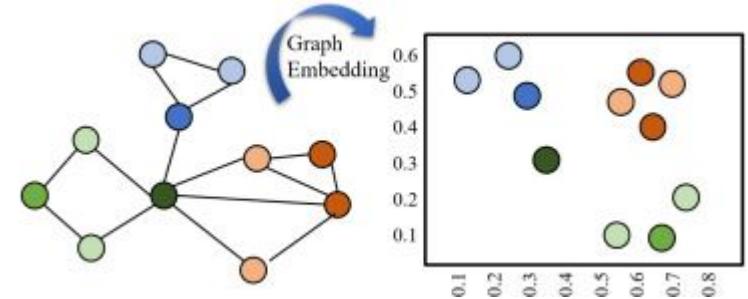
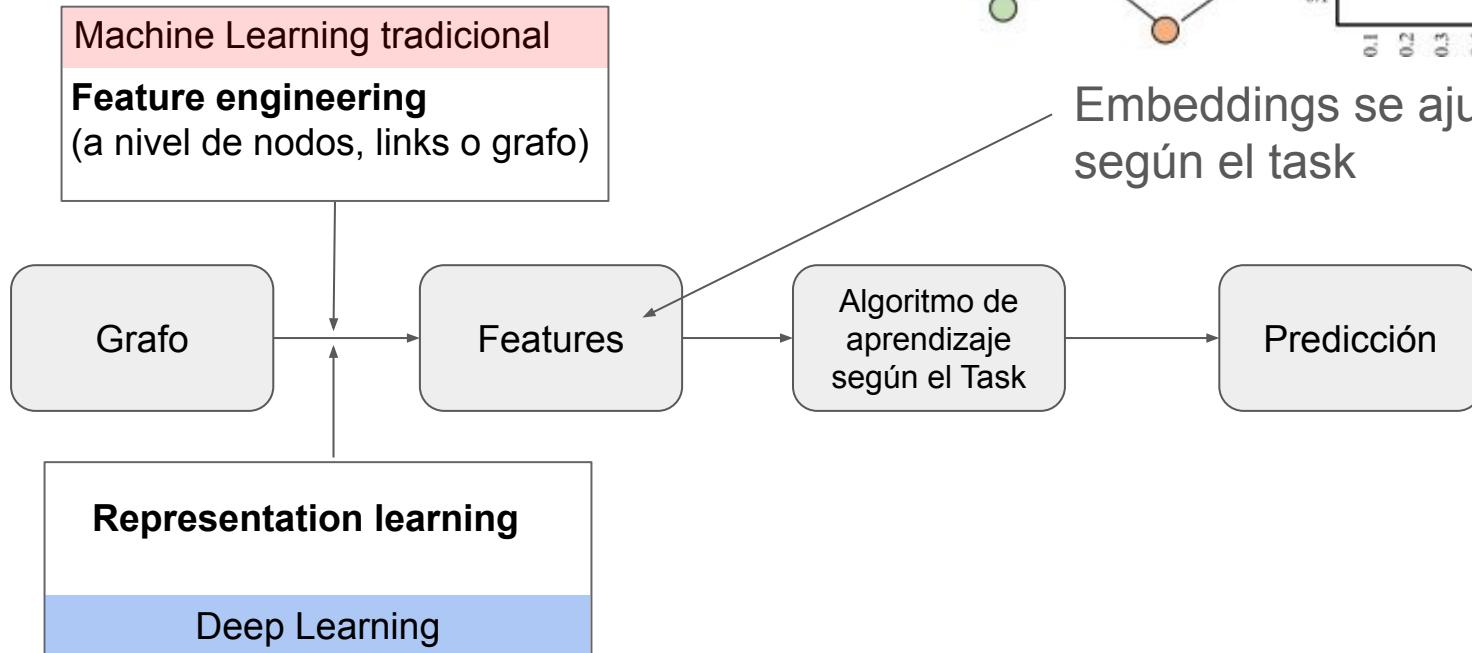
Supervisado



Semi-supervisado

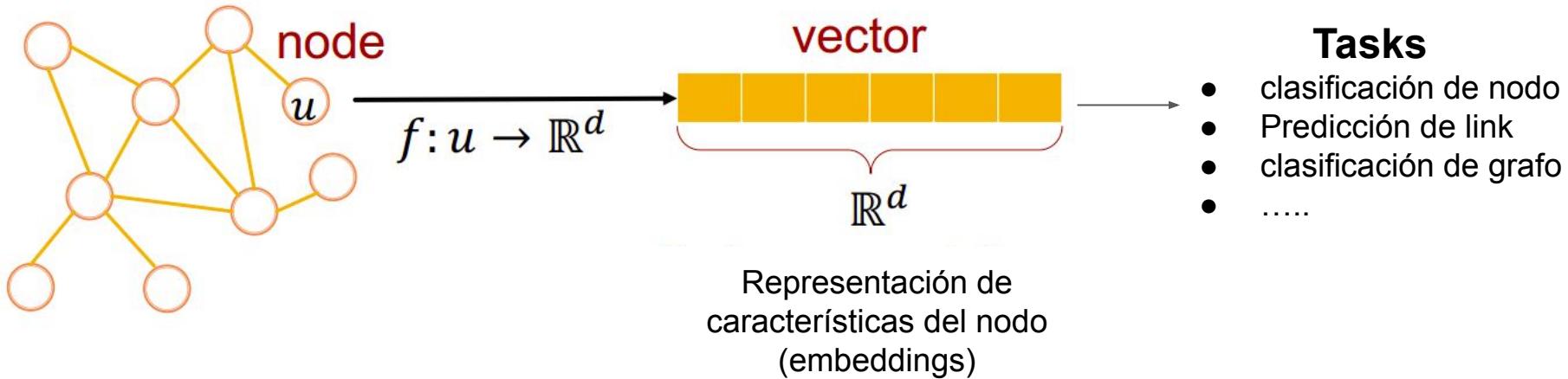


Aprendizaje de representación



Embeddings se ajustan
según el task

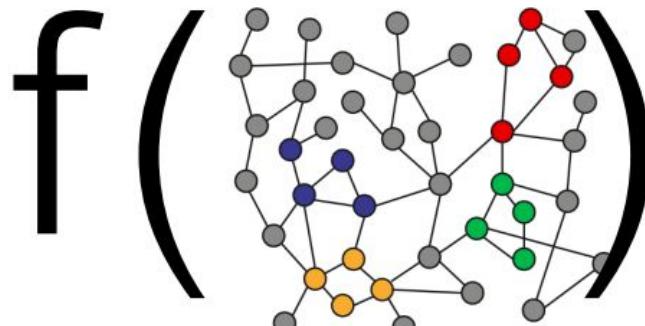
Aprendizaje de representación con grafos



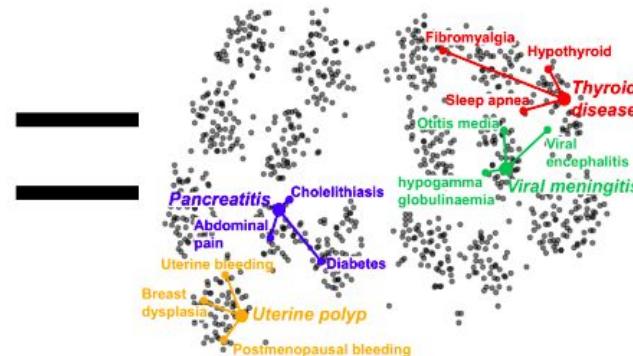
*El aprendizaje debe ser **independiente (agnóstico)** del task posterior*

Como debe ser la función f

| **Objetivo:** encontrar una función f tal que nodos similares en el grafo son mapeados juntos en un embedding d-dimensional



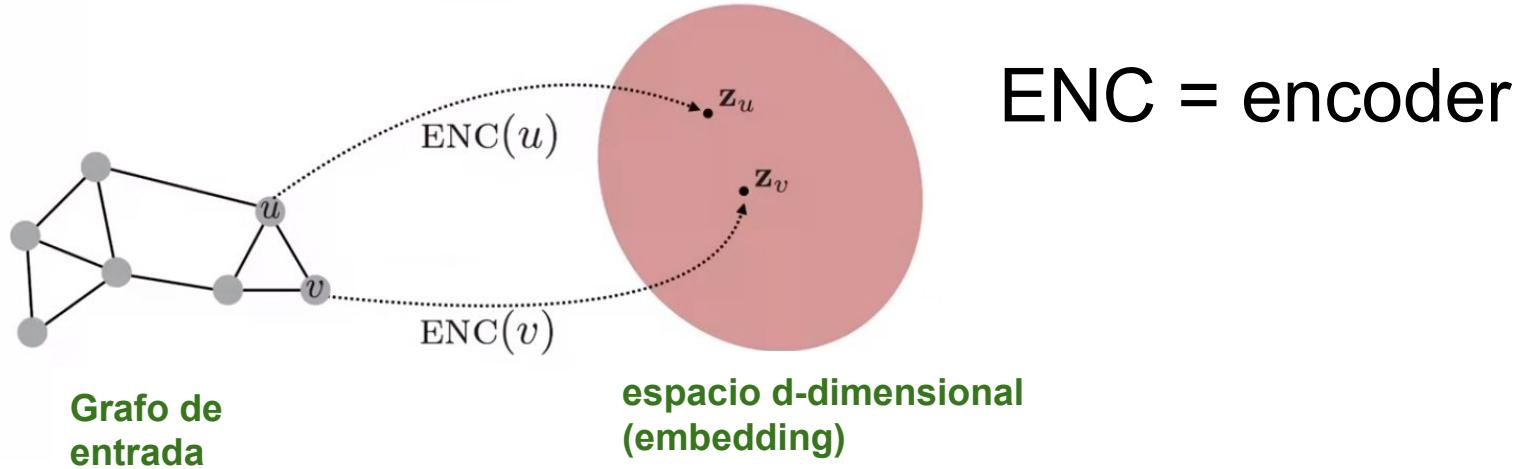
Grafo de entrada



Embedding 2D de los nodos

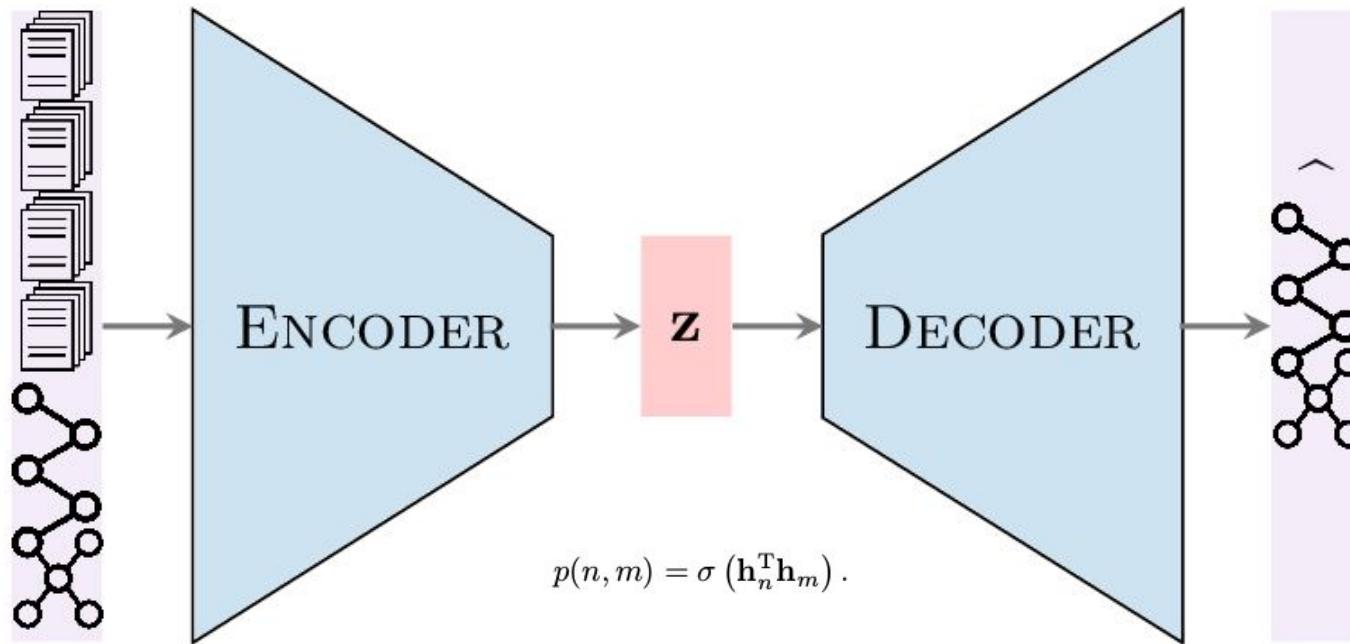
¿Que forma tiene la función f ?

Mapeo sobre el espacio de embeddings

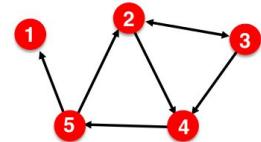


$\text{ENC}(v) =$ **múltiples capas de transformaciones no-lineales a partir de la estructura del grafo**

Autoencoder de Grafo (GAE)



PyTorch Geometric



- Es una extensión de PyTorch que facilita la implementación de GNNs
- Un grafo en PyG se describe mediante una instancia de la clase **torch_geometric.data.Data**

`data = Data(x=x, edge_index=edge_index, y=y)`

con los siguientes atributos:

`data.x` = features de los nodos. Dimension: [num_nodes, num_node_features]

`data.edge_index` = conectividad del grafo en formato COO utilizado para representar matrices dispersas.
Dimension: [2, num_edges]

2	2	3	3	4	5	5	→ origen
3	4	2	4	5	1	2	→ destino

`data.y` = labels del grafo. Dependiendo del task las dimensiones serán

- `[1, *]` a nivel de grafo
- `[num_nodes, *]` a nivel de nodo.

PyG para clasificación de nodos

```
import torch
from torch.nn import Linear
from torch_geometric.nn import GCNConv

class GCN(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = GCNConv(dataset.num_features, hidden)
        self.conv2 = GCNConv(hidden, hidden)
        self.conv3 = GCNConv(hidden, hidden)
        self.classifier = Linear(hidden, 2)

    def forward(self, x, edge_index):
        h = self.conv1(x, edge_index)
        h = h.tanh()
        h = self.conv2(h, edge_index)
        h = h.tanh()
        h = self.conv3(h, edge_index)
        Z = h.tanh()
        out = self.classifier(Z)
        return out, Z
```

GCN simple de Kipf and Welling

$$\mathbf{x}_v^{(\ell+1)} = \mathbf{W}^{(\ell+1)} \sum_{w \in \mathcal{N}(v) \cup \{v\}} \frac{1}{c_{w,v}} \cdot \mathbf{x}_w^{(\ell)}$$

dimensión del embedding

apilamos 3 GCNs (agregamos información de 3 capas de vecinos alrededor de cada nodo)

transformación lineal (mapeo de embeddings sobre las clases de salida)

espacio de embeddings final (Z)

coeficiente de normalización para cada link

la arquitectura de la red se define en el constructor

propagación de las características de entrada (x) a lo largo de la arquitectura