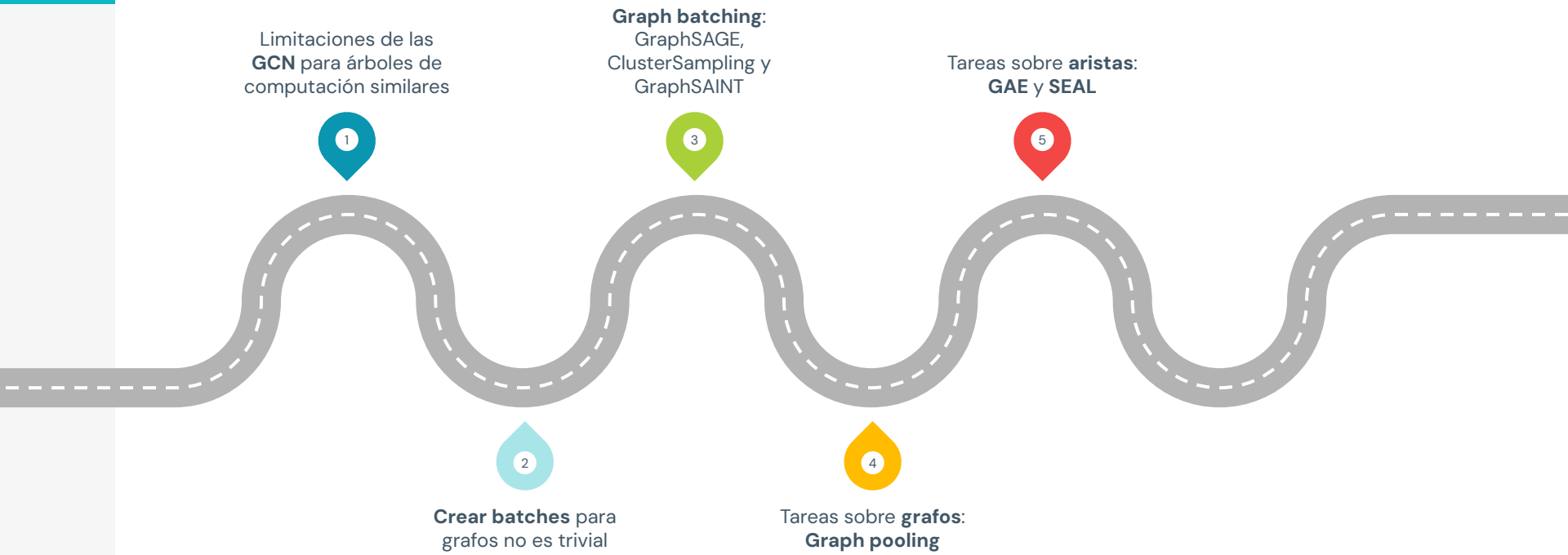
Graph Neural Networks

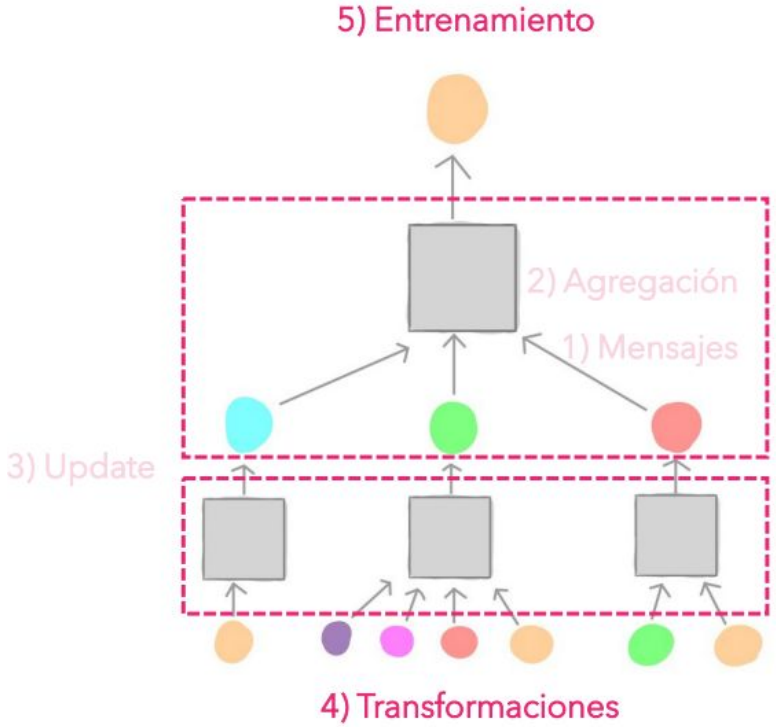
Tema 4 - GNNs Avanzadas

Índice


Limitaciones de las **GCN** para árboles de computación similares
1
Crear batches para grafos no es trivial
2
Graph batching: GraphSAGE, ClusterSampling y GraphSAINT
3
Tareas sobre **grafos**: **Graph pooling**
4
Tareas sobre **aristas**: **GAE** y **SEAL**
5

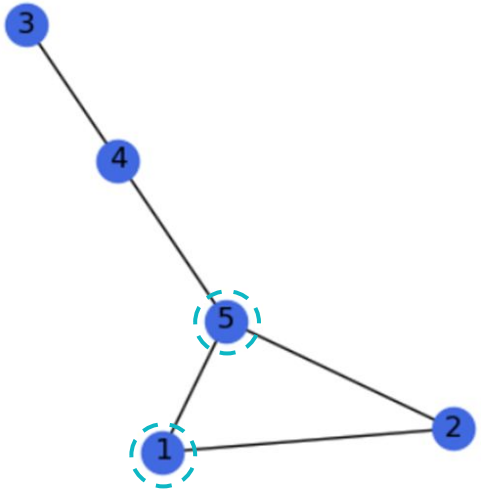
Antes de empezar...

1. **Mensajes –** Cómo transformo los mensajes de los vecinos.
2. **Agregación –** Qué hago para juntar los mensajes.
3. **Update –** Cómo combino los mensajes entre capas.
4. **Transformaciones –** Cambios que se le hacen al grafo antes de entrenar.
5. **Entrenamiento –** Función de pérdida según mi problema y tarea (Nodo, arista, grafo ... etc)

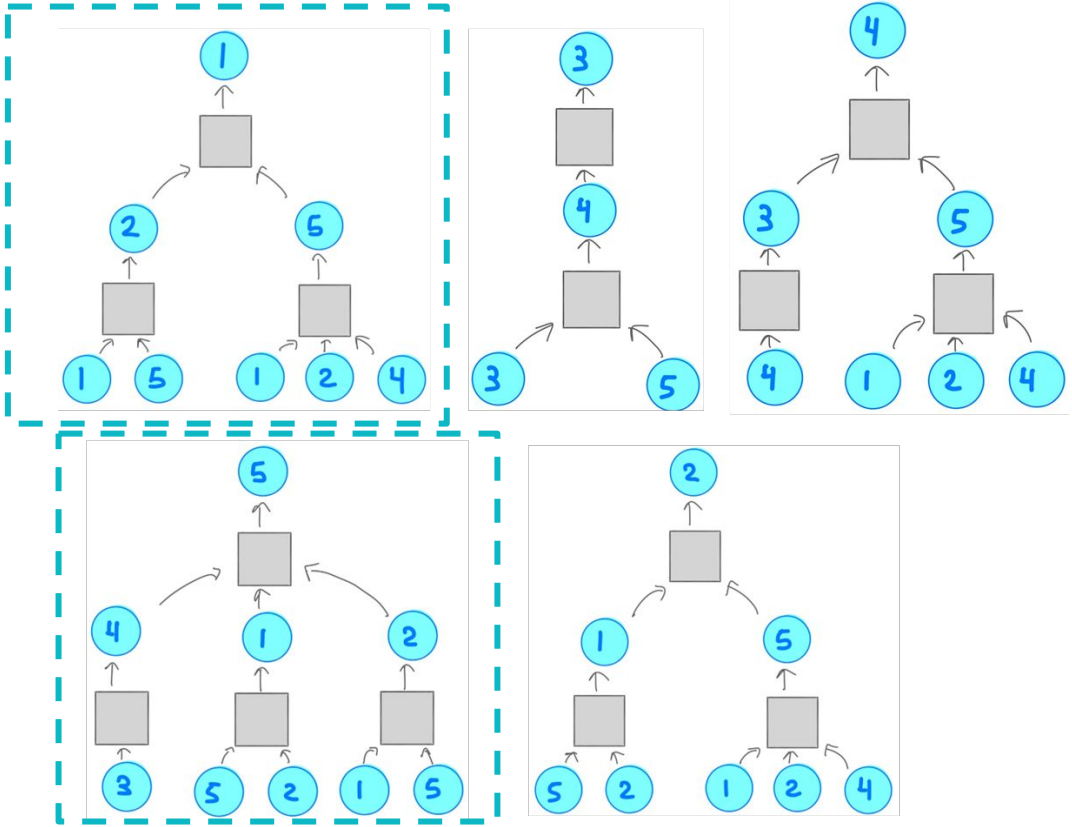


1. Limitaciones de las GCN para árboles de computación similares

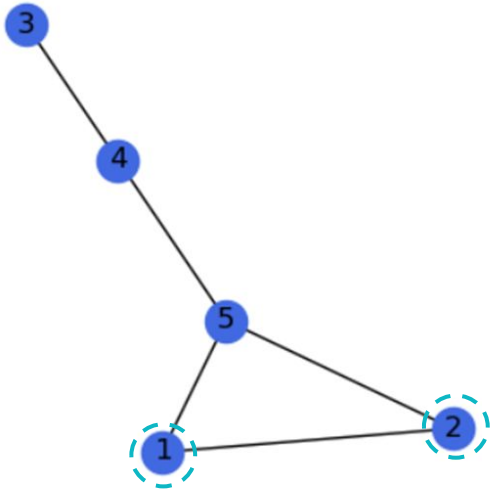
Limitaciones de las GCN



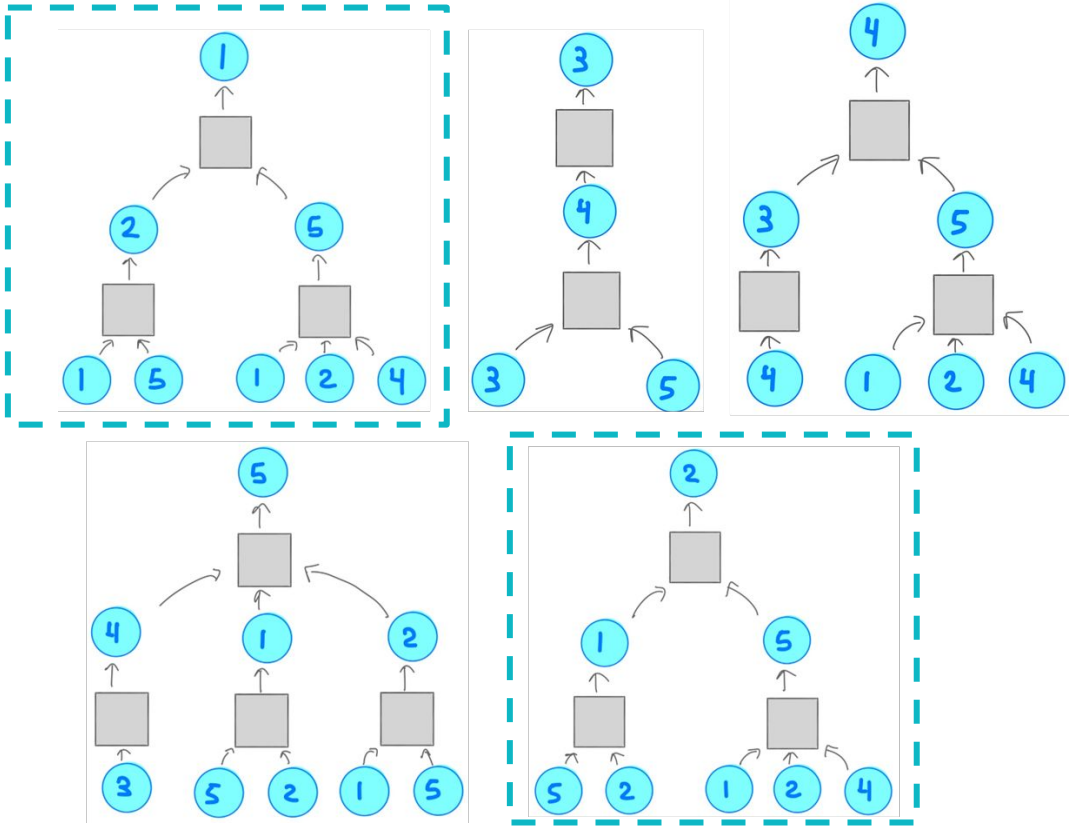
¿Son el nodo **1** y **5** **estructuralmente** diferentes si analizamos sus **árboles de computación** a dos capas de distancia?



Limitaciones de las GCN



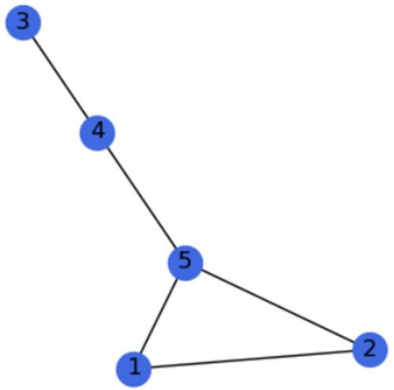
Sin embargo... ¿Son el nodo **1** y **2** **estructuralmente** diferentes si analizamos sus **árboles de computación** a dos capas de distancia?

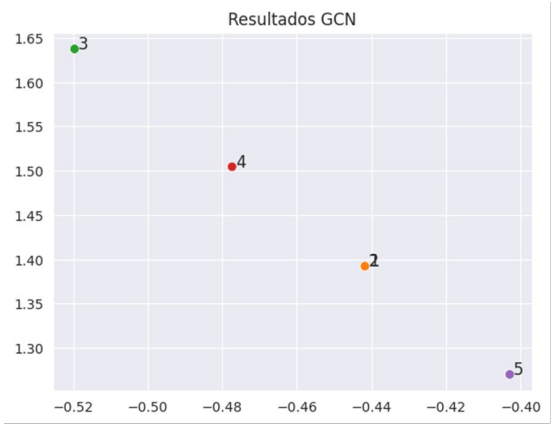


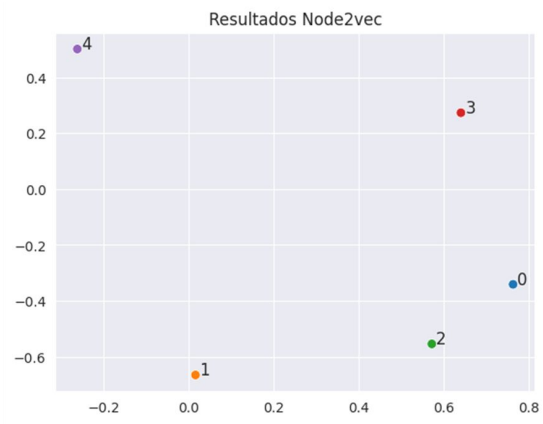
Limitaciones de las GCN

Si no se incorpora información de los **atributos** de los nodos, una **GCN no sería capaz** de diferenciar un par de nodos con los mismos árboles de computación.

En cambio, **alternativas basadas en caminos aleatorios** como Node2Vec sí que **son capaces** de hacer esta distinción ya que capturan patrones globales





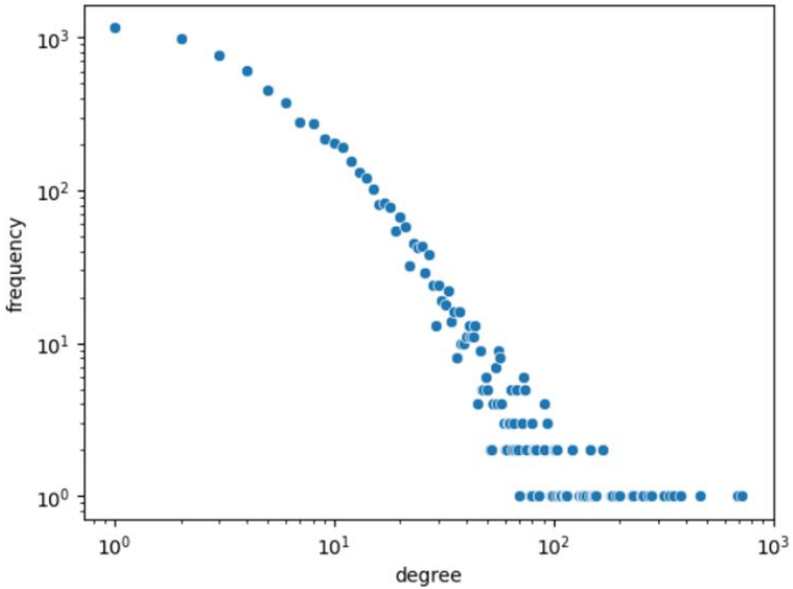


2. Crear batches para grafos no es trivial

¿Por qué batchear un grafo?

Simplemente porque si el grafo tiene muchos nodos **no cabrá en GPU**.

Si existe **un nodo con un grado muy alto**, hacer el paso de mensajes será computacionalmente costoso.



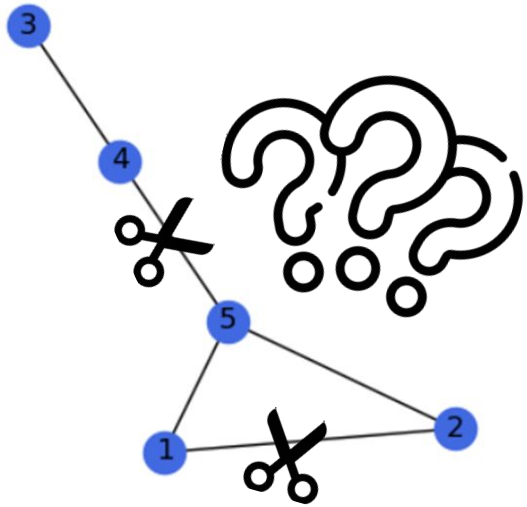
Batchear un grafo no es trivial

Para **datos tabulares** → agrupar varias filas de la tabla.

Para **imágenes** → agrupar varias imágenes.

Para **datos secuenciales** → tomar varias ventanas temporales.

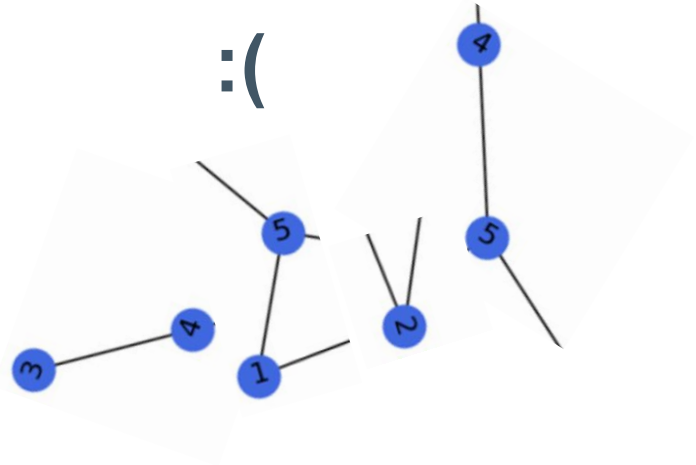
Para **grafos** → **???**



Batchear un grafo no es trivial

Todos los elementos de un grafo **están relacionados entre sí**...
¿Cómo los *separo*?

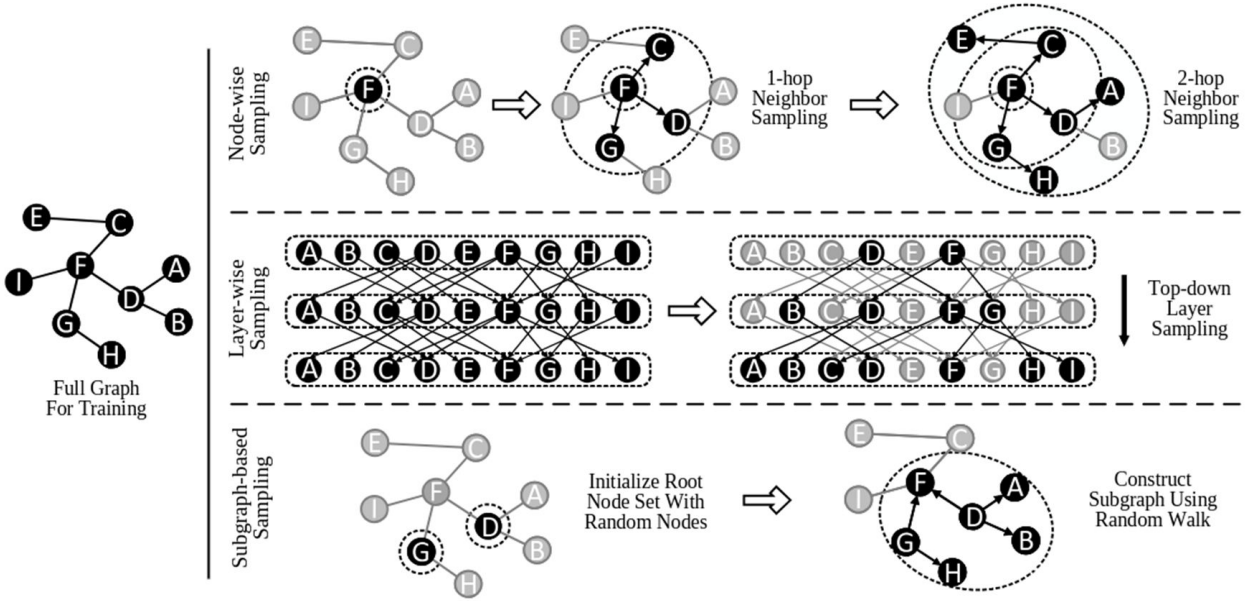
Si elijo solo **ciertos nodos o aristas**... ¿Cómo mantengo la **estructura original** del grafo?
¿Cómo sé que el sub-grafo que obtengo es **equivalente**?



3. Graph Batching

Aproximaciones

Los métodos basados en *capas* no están implementados en PyTorchGeometric!



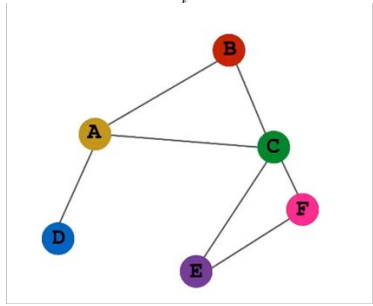
Liu, X., Yan, M., Deng, L., Li, G., Ye, X., & Fan, D. (2021). Sampling methods for efficient training of graph convolutional networks: A survey. *IEEE/CAA Journal of Automatica Sinica*, *9*(2), 205-234.

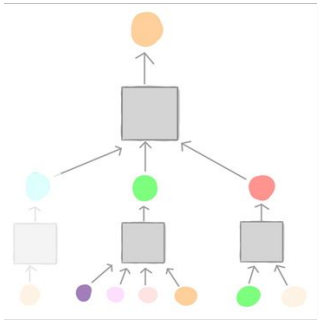
Node-level: GraphSAGE (Neighbor sampling)

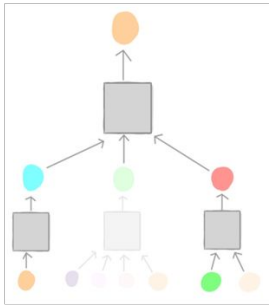
En cada capa **se eligen N vecinos al azar**.

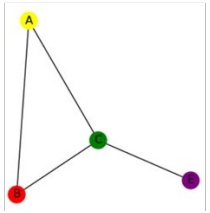
Cada vez que se visita un nodo el árbol de computación cambia.

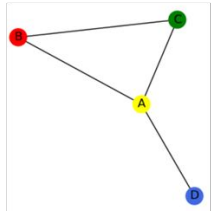
Cada **batch es un subgrafo del original** que incluye los nodos y aristas elegidos durante el sampleo.









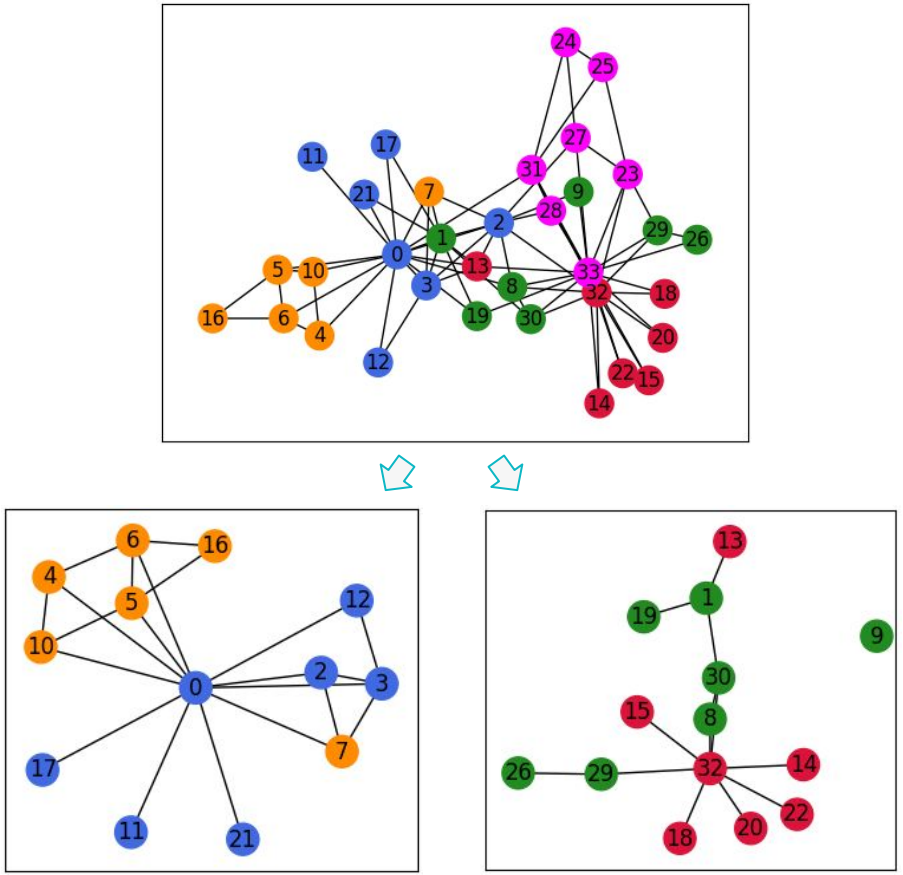


Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems*, *30*.

Graph-level: Cluster Sampling

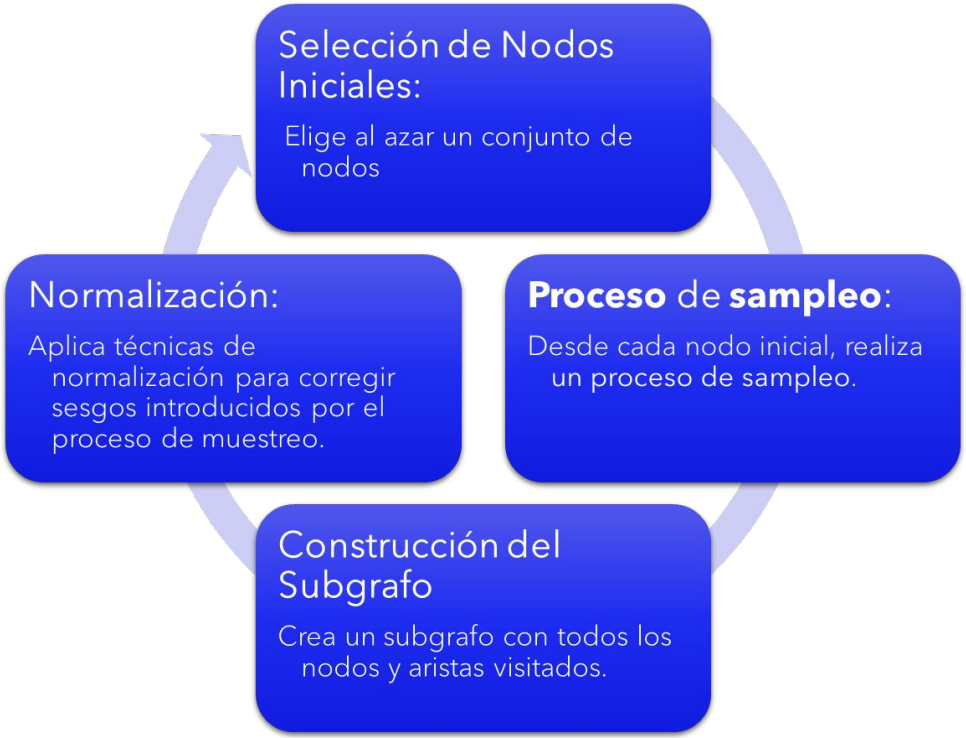
Ejecuta un algoritmo de **detección de comunidades** para dividir el grafo en muchas comunidades relativamente pequeñas.

Cada batch se crea **incluyendo los nodos de N comunidades** y todos los enlaces entre ellos. Tanto intra- como inter-comunidad.



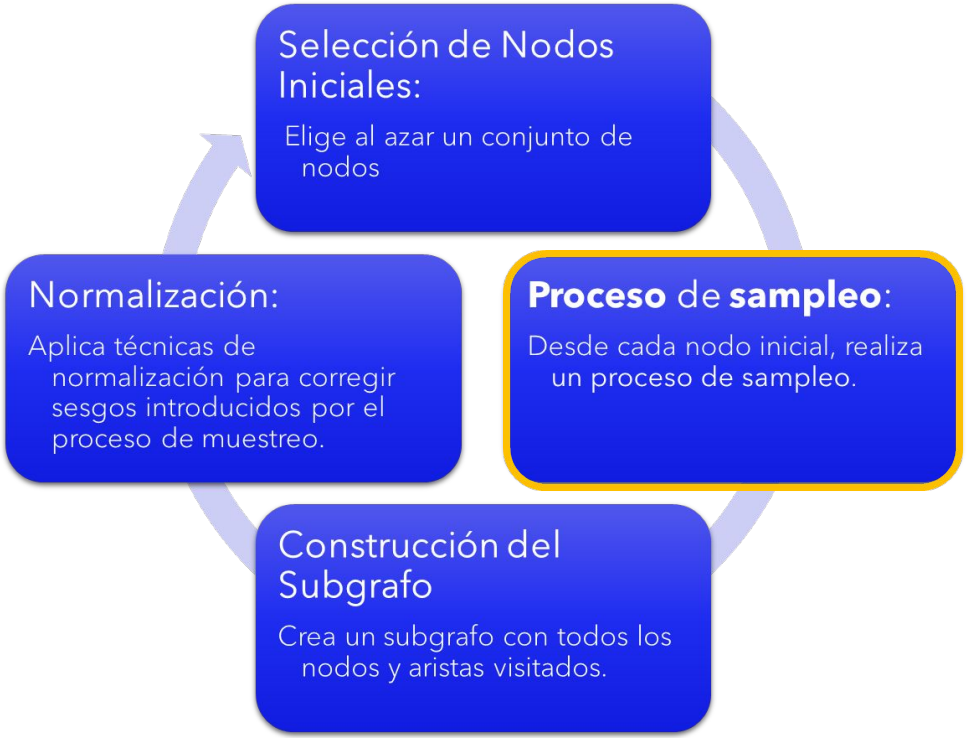
Chiang, W. L., Liu, X., Si, S., Li, Y., Bengio, S., & Hsieh, C. J. (2019, July). Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 257-266).

Graph-level: GraphSAINT



Zeng, H., Zhou, H., Srivastava, A., Kannan, R., & Prasanna, V. (2019). Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*.

Graph-level: GraphSAINT



Node Sampling
Se selecciona un conjunto de nodos aleatoriamente y se extraen todas las aristas asociadas a esos nodos para formar un subgrafo.

Edge Sampling
Se seleccionan bordes aleatoriamente, y los nodos conectados por esos bordes se incluyen en el subgrafo.

Random Walk Sampling
Se realizan caminatas aleatorias para construir subgrafos que capturen mejor las dependencias locales en la estructura del grafo.

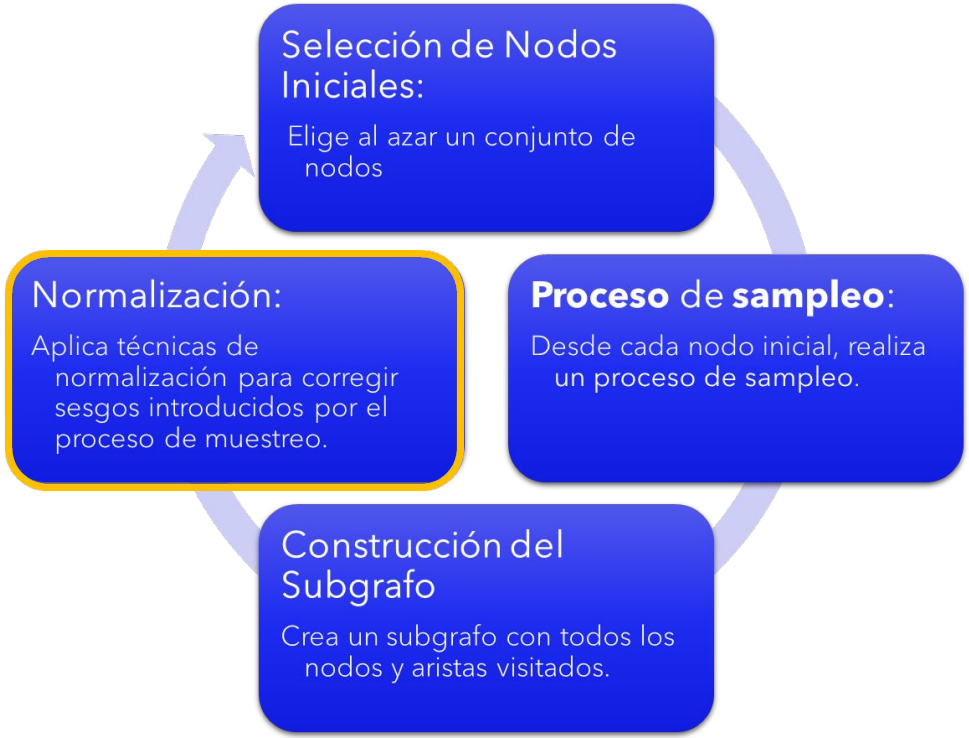
Zeng, H., Zhou, H., Srivastava, A., Kannan, R., & Prasanna, V. (2019). Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*.

Graph-level: GraphSAINT

Para garantizar que las estadísticas del **subgrafo sigan las propiedades del grafo completo** cada nodo o arista en el subgrafo tiene un peso ajustado para compensar la probabilidad de haber sido seleccionado durante el muestreo.

Genera dos coeficientes de corrección:

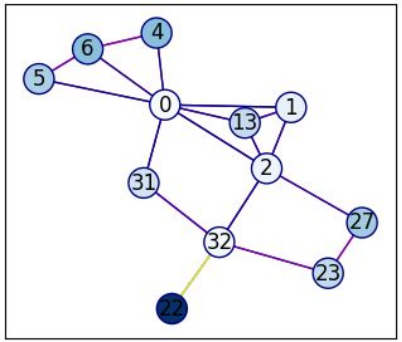
- **α:** para la **agregación** (aristas)
- **λ:** para la **función de pérdida** (nodos)



Zeng, H., Zhou, H., Srivastava, A., Kannan, R., & Prasanna, V. (2019). Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*.

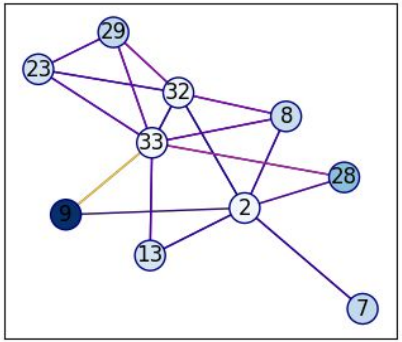
Graph-level: GraphSAINT

Node Sampling



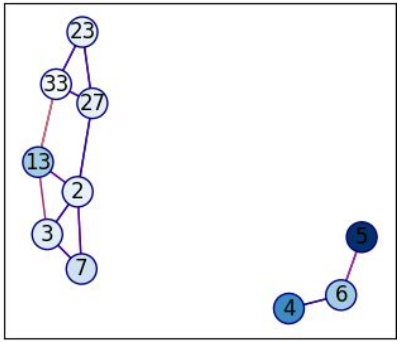
$$P(u) \propto \left\| \tilde{A}_{:,u} \right\|^2$$

Edge Sampling



$$P(e_{u,v}) \propto \frac{1}{\deg(u)} + \frac{1}{\deg(v)}$$

RW Sampling



Desde *r* nodos, random walks de longitud *h*

Según la visualización, acertar un nodo más *claro* premiaría menos que acertar uno más *oscuro* (**λ**).

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., & Prasanna, V. (2019). Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*.

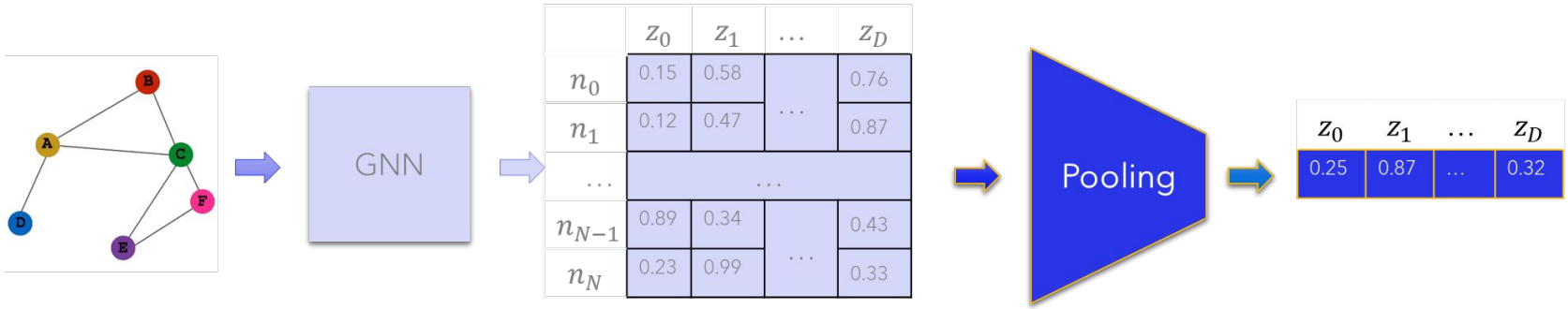
Comparativa

Método	GraphSAGE	Cluster Sampling	GraphSAINT
Eficiencia	**Moderada**, se computa cada vez que se forma el batch.	**Alta**, el algoritmo de comunidades se computa una única vez al comienzo del entrenamiento, el resto es selección aleatoria de clústers una vez por época.	**Moderada-Alta**, se computa al comienzo de cada época.
Generalización	**Limitada** si los nodos clave no se incluyen en el batch.	**Alta** generalización para grafos densos, pudiendo **fallar** en regiones dispersas.	**Alta** generalización debido a la corrección probabilística de los subgrafos.
Ventajas	**Sencillo** y eficiente para tareas **donde la localidad es clave.**	Eficiente para captar patrones **locales en grafos densos**.	Representa bien tanto características **locales como globales**.
Limitaciones	Puede **fallar en captar patrones globales** en grafos especialmente **grandes**.	Depende fuertemente del algoritmo de **detección de comunidades**.	Depende del diseño correcto de las **probabilidades de muestreo**.

4. Tareas sobre grafos

Capas de pooling

Para trabajar a nivel de grafo debemos añadir una **capa de pooling** que transforme los embeddings de los nodos en único vector.



Cualquiera de las vistas en clase

Embeddings de cada nodo

Capa pooling

Embedding del grafo

Función de pooling

Existen múltiples maneras de realizar **graph pooling**.

La función de pooling puede ser **cualquier transformación sobre los embeddings de los nodos** que cumpla:

1. $f_{pool}(\{z_v \in \mathbb{R}^d \mid v \epsilon V\}) = z_G \epsilon \mathbb{R}^d$ ([num_nodes, hidden_dim] a [hidden_dim]).
2. Que f_{pool} sea **invariante al orden**

Trivialmente, operaciones como la **suma**, la **media**, el **máximo**, ..., podrían valernos!

Graph Pooling: RNN y atención

Podemos aplicar el **algoritmo de atención sobre los embedding de los nodos** para estudiar la ***influencia* que unos tienen sobre otros**.

Utilizaremos el **mecanismo de actualización propio de las RNN** para conseguir las *queries* que se irán propagando durante *T* iteraciones.

La salida final resultará de la **concatenación de las salidas** del mecanismo de actualización.

Graph Pooling: RNN y atención

1. Calculamos la nueva *query* a partir de la antigua salida de atención y de la antigua *query* (entendemos *query* como *hidden state*) con una red de tipo RNN.
2. Calculamos la similitud entre el embedding de cada nodo y la nueva *query* (cualquier función $\mathbb{R}^d \rightarrow \mathbb{R}$ nos vale, normalmente usaremos el producto escalar).
3. Normalizamos las similitudes con una *softmax* obteniendo los *attention scores*.
4. Se multiplica cada *attention score* ([0, 1]) por el valor original del embedding del nodo **ponderando su *importancia* en el contexto del grafo**. La nueva salida será la suma de cada embedding ponderado.
5. Se repite del paso 1 al 4 *T* veces y finalmente se concatenan las salidas obteniendo el embedding del grafo.

$$q_t = RNN(o_{t-1}, q_{t-1})$$

o_0 y q_0 se inicializan a 0!

$$e_{u,t} = f_\alpha(z_u, q_t)$$

$$\alpha_{u,t} = \frac{\exp(e_{u,t})}{\sum_{v \in V} \exp(e_{v,t})}$$

$$o_t = \sum_{u \in V} \alpha_{u,t} z_u$$

$$Z_G = (o_1 \oplus o_2 \oplus \cdots \oplus o_T)$$

5. Tareas sobre aristas

Predicción de aristas

Técnicas basadas en nodos

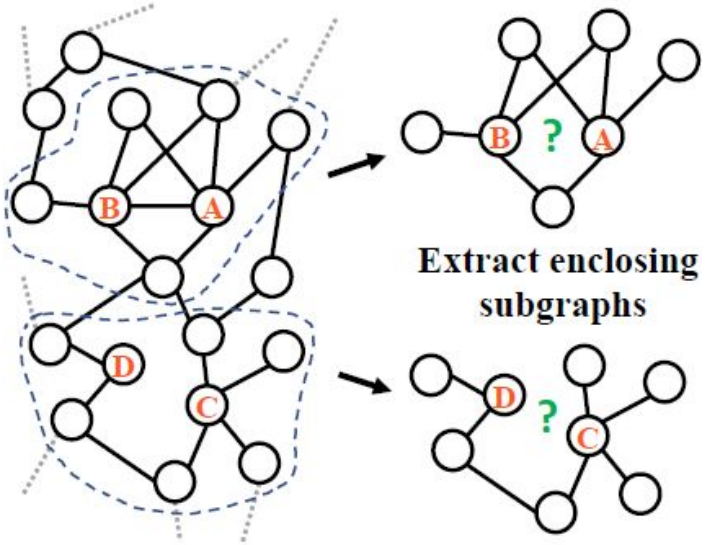
- Uso una GNN para calcular los **embeddings** de cada nodo.
- Se utilizan **pares de nodos** como representaciones de las aristas.

Técnicas basadas en subgrafos

- Creo un **subgrafo a partir de una pareja de nodos**.
- Genero **etiquetas** para los nodos del subgrafo.
- Entreno una GNN para procesar el subgrafo e identificar si hay arista o no.

SEAL

1. Sampleo parejas de nodos.
2. Creo un subgrafo para esa pareja de nodos.
3. Etiqueto los nodos del subgrafo.
4. Elimino la arista que conecta la pareja de nodos.
5. Entreno una GNN para predecir si falta o no la arista (a nivel de grafo).



Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. *Advances in neural information processing systems*, *31*.

Graph Neural Networks

Tema 2 - Shallow Embeddings

UNIVERSIDAD POLITÉCNICA DE MADRID

Máster
Deep Learning