



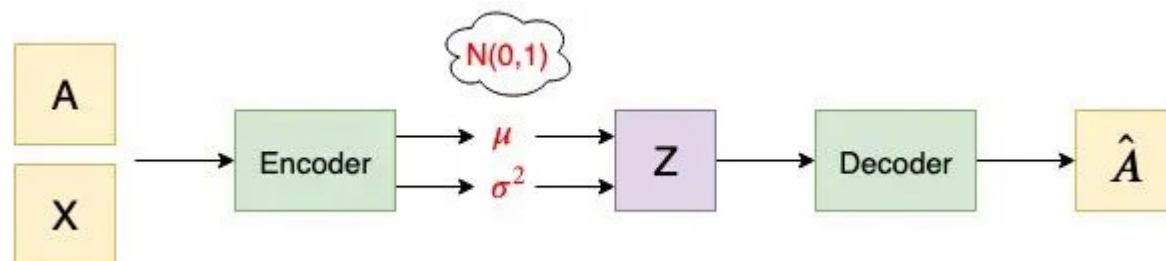
IIC-3641 Aprendizaje Automático basado en Grafos

<https://github.com/marcelomendoza/IIC3641>

- RECAPITULACIÓN -

Variational Graph Autoencoder

En la AF8 trabajamos con el Variational Graph Autoencoder (VGAE), un modelo que permite hacer autoencoding de grafos.



La implementación con la que trabajamos del VGAE dejaba la matriz A en el constructor, es decir, esta implementación tenía una estructura fija. En forward lo que hacíamos era colocar el X , es decir, el VGAE nos permitía codificar, en base a una estructura fija, los nodos del grafo.

Con este modelo pudimos reconstruir la matriz de adyacencia y trabajar con los node embeddings en tareas de clasificación de nodos.

Es un modelo interesante ya que, en base a una estructura fija, podemos recuperar subgrafos de A haciendo forward de algunos nodos. Esto es interesante ya que, por ejemplo, el A podría describir un grafo de referencia, y podríamos estar interesados en reconstruir subgrafos.



Kipf, T., Welling, M. Variational Graph Auto-Encoders, NeurIPS 2016.
<https://arxiv.org/abs/1611.07308>

Variational Graph Autoencoder

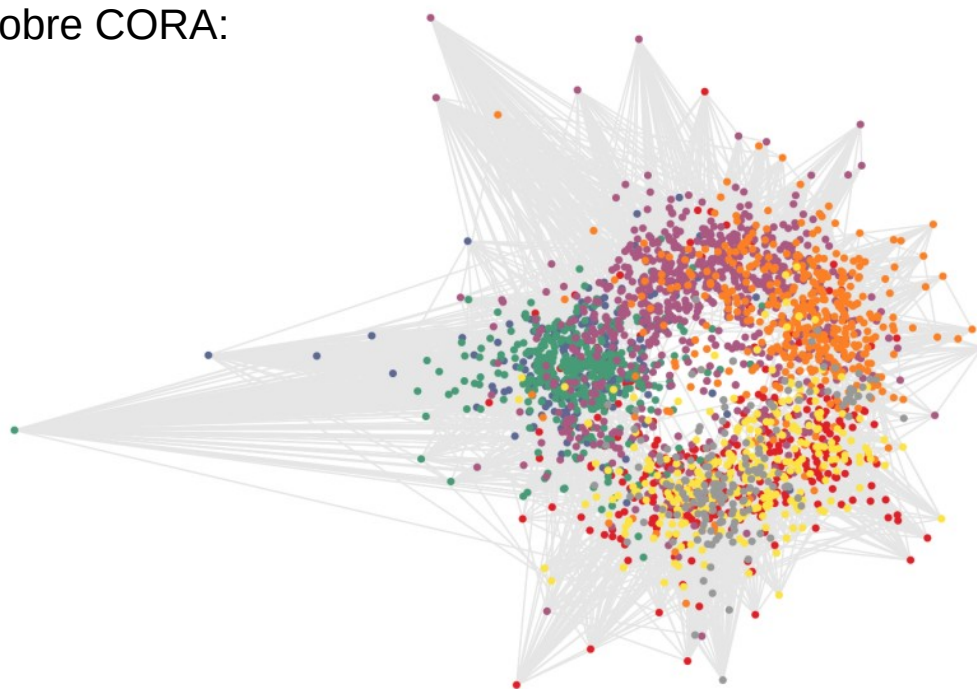
¿Qué aprende la VGAE?

Miremos que es lo que reconstruye:

$$p(A_{ij} = 1 \mid \mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^\top \mathbf{z}_j)$$

Para que la función logística se active, el \mathbf{z}_i y \mathbf{z}_j deben estar cerca en el espacio latente. Si trabajamos con \mathbf{Z} , lo que reconstruye es un grafo.

Veamos la VGAE sobre CORA:



La VGAE es un learner no supervisado de **representaciones de nodos**, ya que lo que hace forward es el \mathbf{X} . Usando estas representaciones, podemos reconstruir subgrafos, es decir, combinaciones de nodos, midiendo que tan probable es que estos nodos se enlacen.

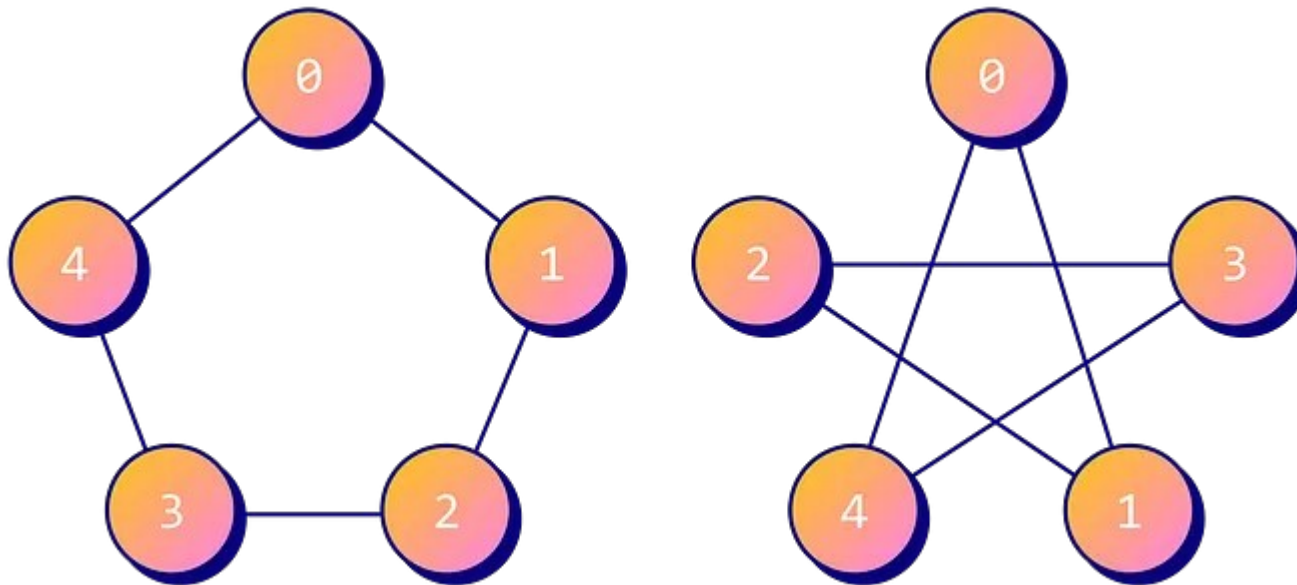
- GRAPH ISOMORPHISM NETWORKS -

¿Cuán expresivos son los modelos que hemos visto?

Algo relevante en GML consiste en dilucidar si los modelos que hemos visto tienen la capacidad de distinguir grafos que estructuralmente son equivalentes. La idea es que la expresividad de una red neuronal de grafo está relacionada con su capacidad de distinguir isomorfismos de grafos.

¿Qué es un isomorfismo de grafo?

Existe un isomorfismo entre dos grafos si tienen la misma estructura, esto es, tienen idénticas conexiones pero una permutación en el etiquetado de los nodos.



Existe un test denominado test de Weisfeiler-Lehman el cual nos puede decir si dos grafos **no son isomórficos**. El test no da garantías de isomorfismo pero si de no isomorfismo.

Test de Weisfeiler-Lehman

Veamos algunas definiciones.

Isomorfismo de grafos: Dos grafos son considerados isomórficos si existe un mapping entre los nodos de ambos grafos que preserve la adyacencia de los nodos.

Test de isomorfismo: problema difícil, no se conoce un algoritmo de tiempo polinomial que determine si dos grafos son isomórficos. Tampoco se sabe si es NP-completo. Se dice que es un ejemplo de problema NP-intermedio.

The Weisfeiler-Lehman test: se basa en la reducción a la forma canónica de ambos grafos. Si ambos grafos difieren en sus formas canónicas, entonces no son isomórficos.

El reverso no es cierto. Dos grafos no isomórficos podrían compartir la misma forma canónica. Es decir, si el test reduce ambos grafos a la misma forma canónica, no podemos decir nada.

Test de Weisfeiler-Lehman

Test (invariante para obtener forma canónica)

- En la iteración i del test asignaremos a cada nodo una tupla $L_{i,n}$ que contiene las etiquetas que el nodo ha tenido en las iteraciones anteriores (*compressed label*) y un multiset¹ con los *compressed labels* de sus vecinos.
- En cada iteración i , a cada nodo n se le asignará un nuevo compressed label $C_{i,n}$.

Algoritmo para obtener forma canónica

1. Comenzamos inicializando con $C_{0,n} = 1$ a todos los nodos.
2. En la iteración i del algoritmo, para cada nodo n , asignamos a $L_{i,n}$ la tupla que contiene las etiquetas anteriores del nodo, es decir, $C_{i-1,n}$, y el multiset $C_{i-1,m}$, donde m se mueve entre los vecinos de n .
3. Completamos la iteración i haciendo que $C_{i,n}$ sea un nuevo compressed label, usando el hash $L_{i,n}$. Todo par de nodos con el mismo $L_{i,n}$ debe provenir de un mismo $C_{i,n}$.
4. Particionaremos los nodos del grafo según su *compressed label*. Terminamos si $i = N$ o bien si no hay cambios en la partición de nodos entre dos iteraciones consecutivas.

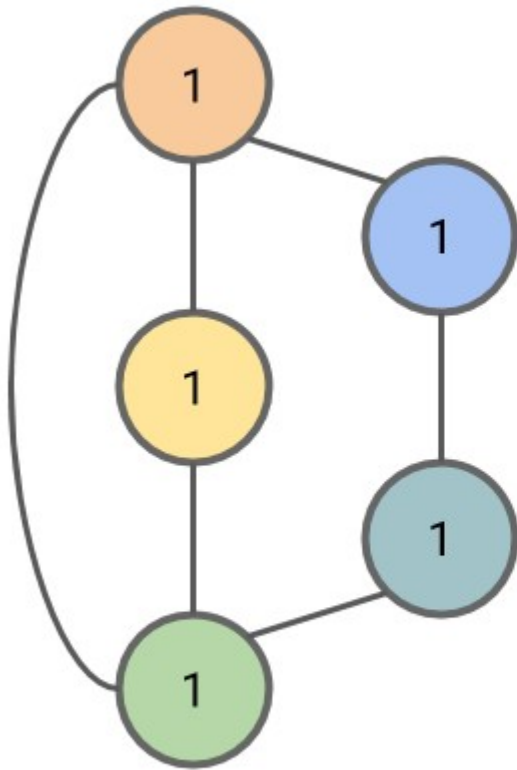
¹ Un multiset es un conjunto en el cual se pueden repetir los elementos

Test de Weisfeiler-Lehman

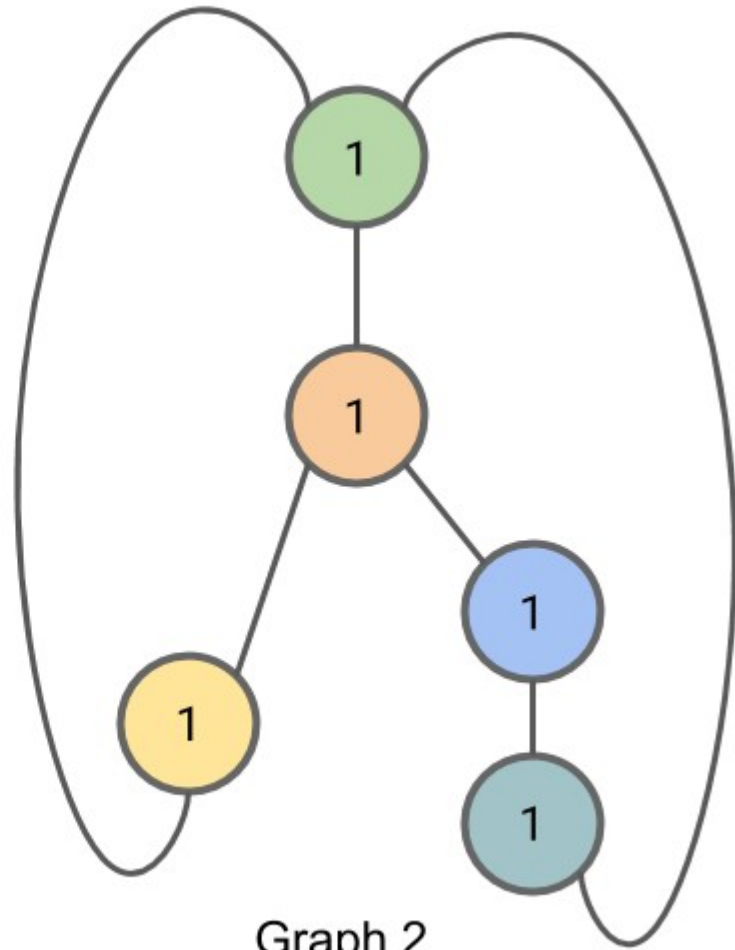
Algoritmo para obtener forma canónica

1. Comenzamos inicializando con $C_{0,n} = 1$ a todos los nodos.

C_0



Graph 1



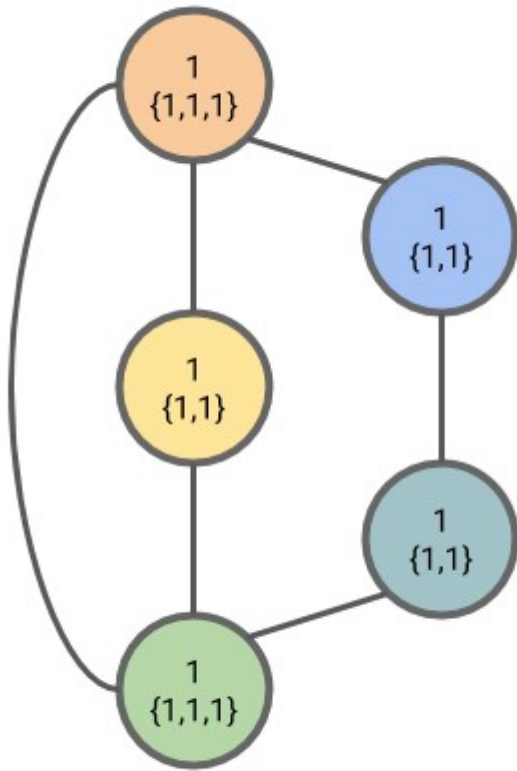
Graph 2

Test de Weisfeiler-Lehman

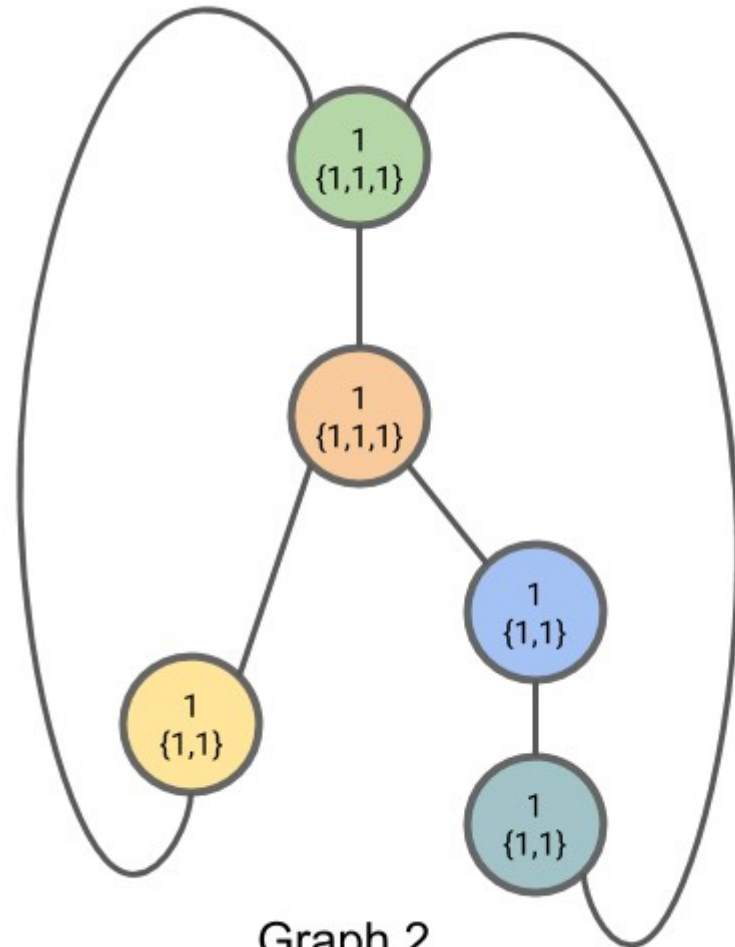
Algoritmo para obtener forma canónica

2. En la iteración 1 del algoritmo, para cada nodo n , asignamos a $L_{i,n}$ la tupla que contiene las etiquetas anteriores del nodo, es decir, $C_{i-1,n}$, y el multiset $C_{i-1,m}$, donde m se mueve entre los vecinos de n .

L_1



Graph 1



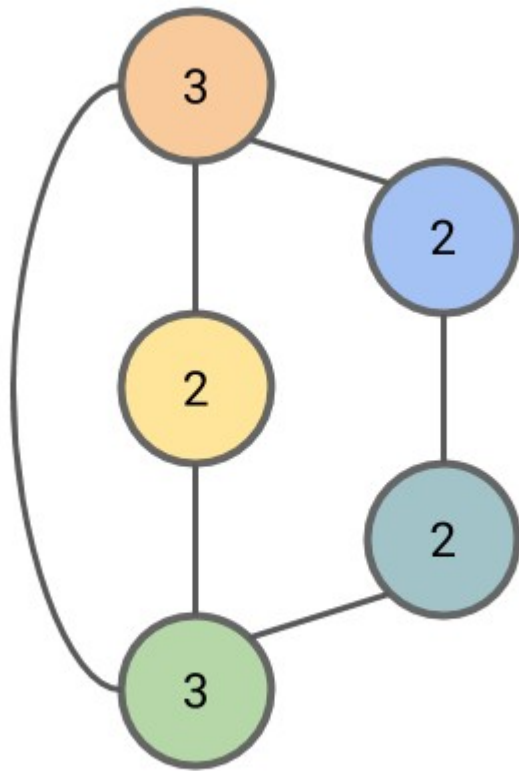
Graph 2

Test de Weisfeiler-Lehman

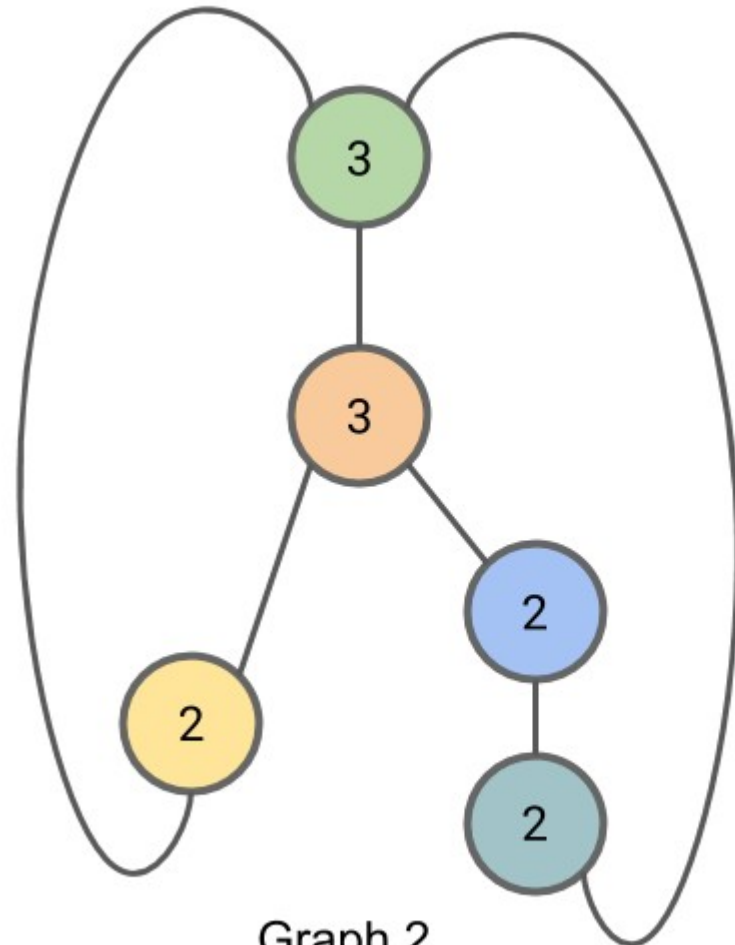
Algoritmo para obtener forma canónica

3. Completamos la iteración 1 haciendo que $C_{i,n}$ sea un nuevo compressed label, usando el hash $L_{i,n}$. Todo par de nodos con el mismo $L_{i,n}$ debe provenir de un mismo $C_{i,n}$.

C_1



Graph 1



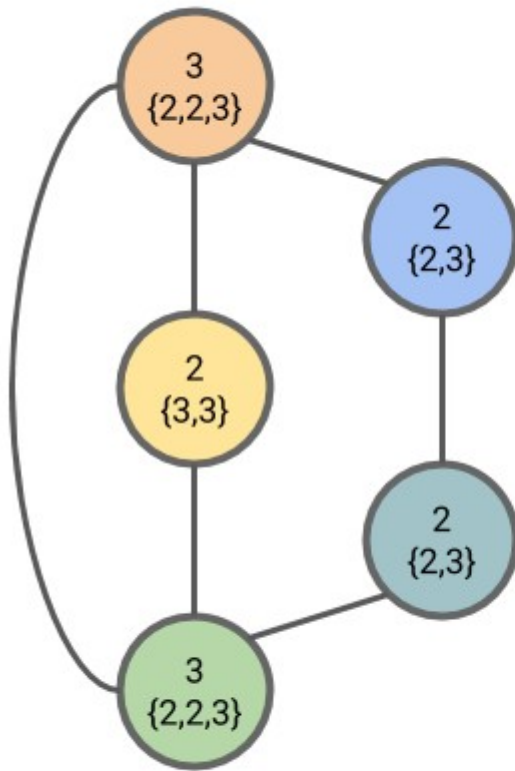
Graph 2

Test de Weisfeiler-Lehman

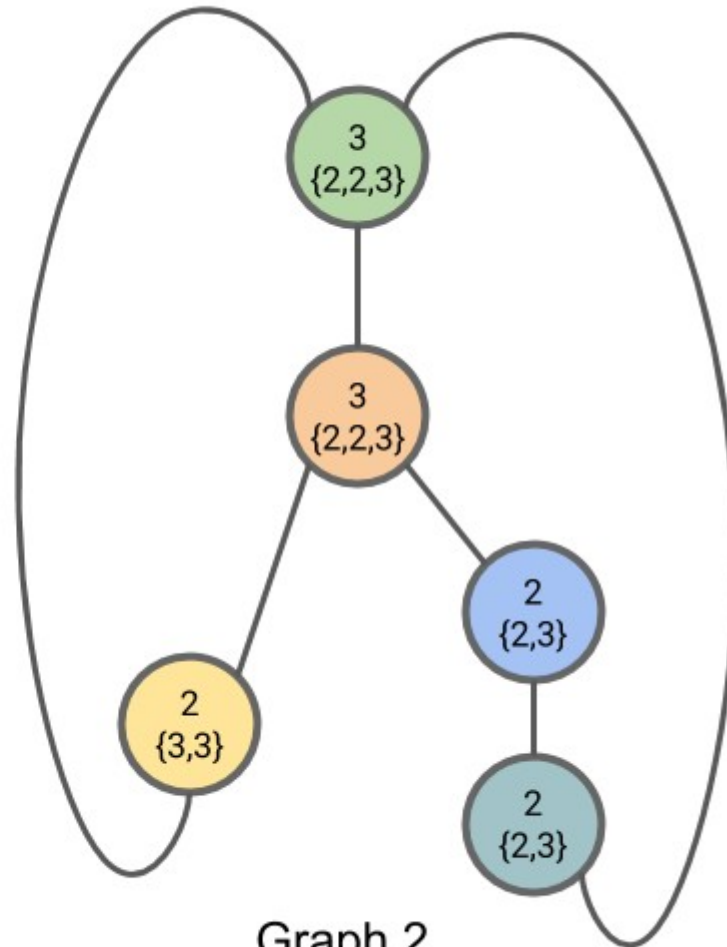
Algoritmo para obtener forma canónica

2. En la iteración 2 del algoritmo, para cada nodo n , asignamos a $L_{i,n}$ la tupla que contiene las etiquetas anteriores del nodo, es decir, $C_{i-1,n}$, y el multiset $C_{i-1,m}$, donde m se mueve entre los vecinos de n .

L_2



Graph 1



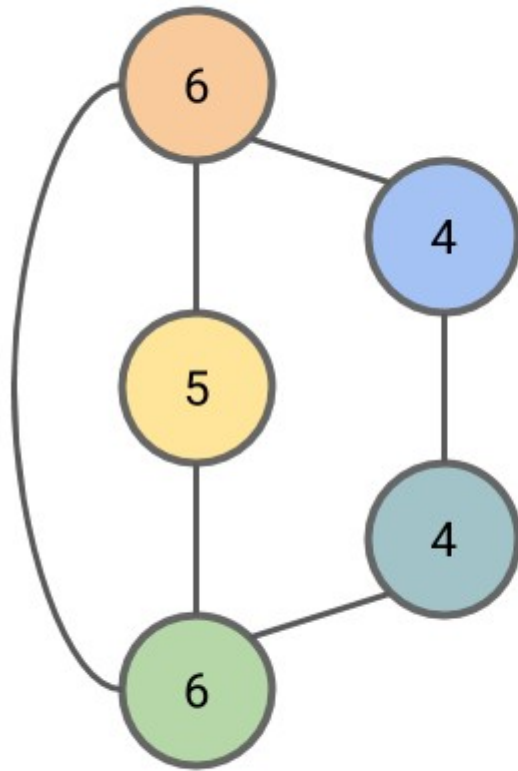
Graph 2

Test de Weisfeiler-Lehman

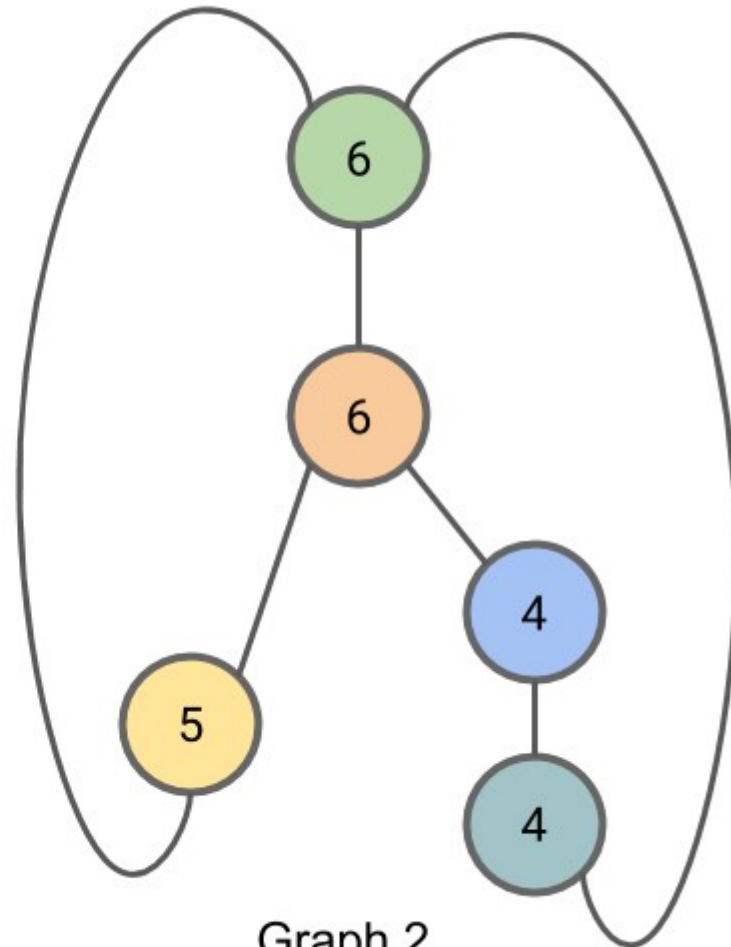
Algoritmo para obtener forma canónica

3. Completamos la iteración 2 haciendo que $C_{i,n}$ sea un nuevo compressed label, usando el hash $L_{i,n}$. Todo par de nodos con el mismo $L_{i,n}$ debe provenir de un mismo $C_{i,n}$.

C_2



Graph 1



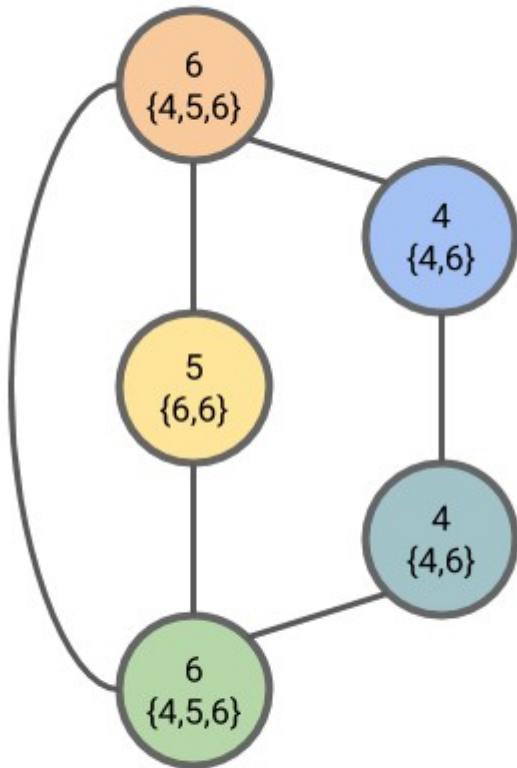
Graph 2

Test de Weisfeiler-Lehman

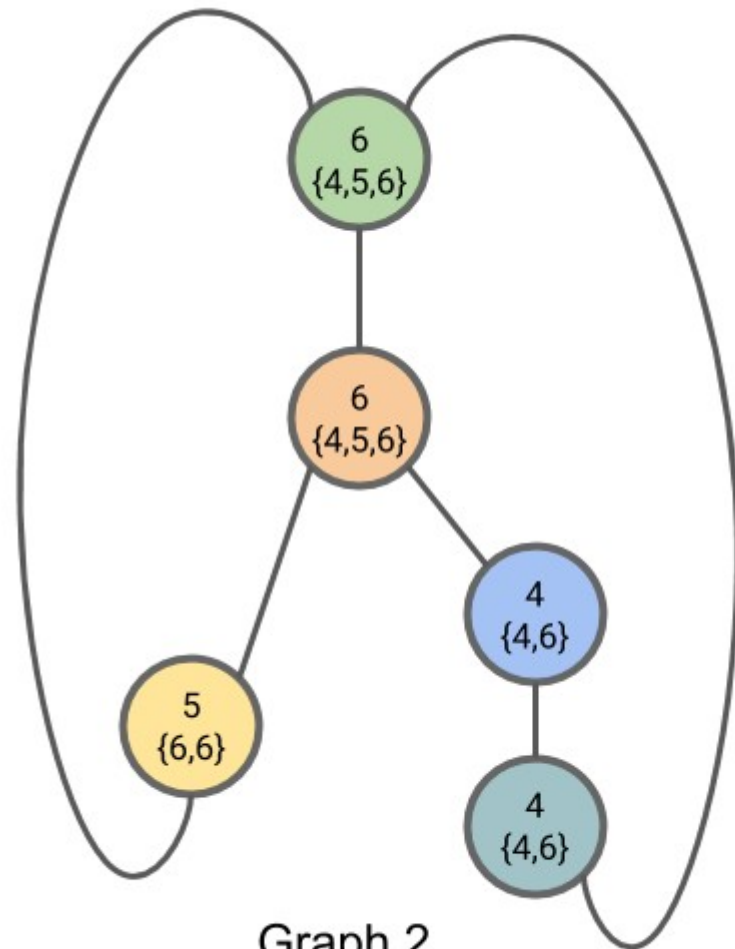
Algoritmo para obtener forma canónica

2. En la iteración 3 del algoritmo, para cada nodo n , asignamos a $L_{i,n}$ la tupla que contiene las etiquetas anteriores del nodo, es decir, $C_{i-1,n}$, y el multiset $C_{i-1,m}$, donde m se mueve entre los vecinos de n .

L_3



Graph 1



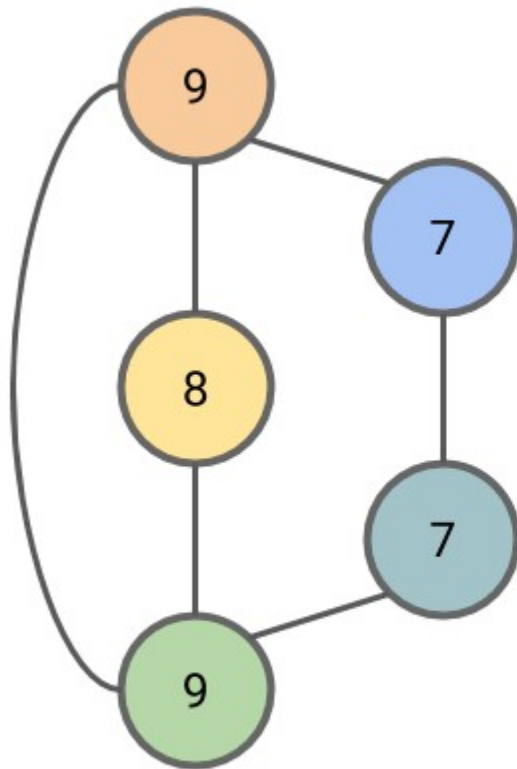
Graph 2

Test de Weisfeiler-Lehman

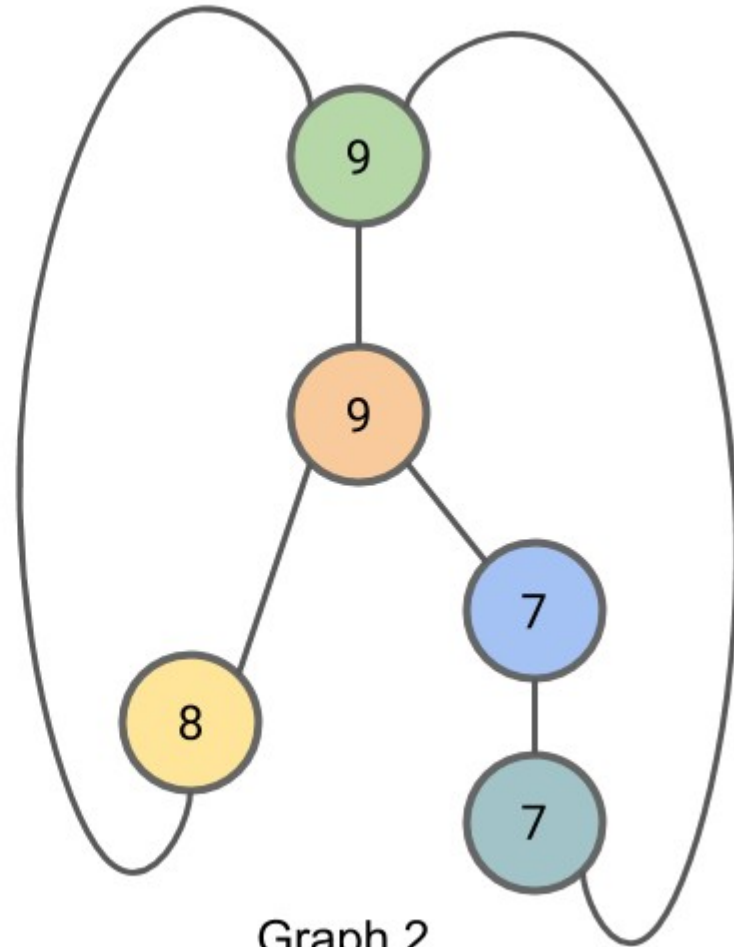
Algoritmo para obtener forma canónica

3. Completamos la iteración 3 haciendo que $C_{i,n}$ sea un nuevo compressed label, usando el hash $L_{i,n}$. Todo par de nodos con el mismo $L_{i,n}$ debe provenir de un mismo $C_{i,n}$.

C_3



Graph 1



Graph 2

Observemos que no han habido cambios entre C_2 y C_3 , entonces el algoritmo termina. Observamos que los mappings son equivalentes: 2 7s, 1 8, 2 9s.

Graph Isomorphic Networks (GIN)

Las redes GIN usan un agregador a nivel de nodos que se basa en la idea del WL test. El efecto de usar un agregar basado en el WL test es que la red produzca embeddings de nodos distintos si los grafos no son isomorfos.

Seguiremos el razonamiento del paper. Las condiciones para que una GNN mapee a embeddings distintos dos grafos G_1 y G_2 que el test WL decide que no son isomórficos son:

a) La red calcula iterativamente los embeddings de los nodos según:

$$h_v^{(k)} = \phi \left(h_v^{(k-1)}, f \left(\left\{ h_u^{(k-1)} : u \in \mathcal{N}(v) \right\} \right) \right)$$

donde tanto f como ϕ deben ser funciones inyectivas.

b) El graph read-out de la red, que opera sobre un multiset de nodos $\left\{ h_v^{(k)} \right\}$, debe ser inyectivo.



Xu, K., Hu, W., Leskovec, J., Jegelka, S. How powerful are graph neural networks?, ICLR 2019.
<https://arxiv.org/pdf/1810.00826v3>

Graph Isomorphic Networks (GIN)

Asumamos que el espacio de características de la entrada \mathcal{X} es contable. Esto es importante ya que debemos estudiar la inyectividad y en estos espacios esta propiedad está bien caracterizada.

Lema (ver demo en paper): Existe una función $f : \mathcal{X} \rightarrow \mathbb{R}^n$ tal que para infinitas elecciones de epsilon, se cumple que:

$$h(c, X) = (1 + \epsilon) \cdot f(c) + \sum_{x \in X} f(x)$$

es único para cada par (c, X) , donde $c \in \mathcal{X}$ y $X \subset \mathcal{X}$ es un multiset de tamaño acotado. Más aún, cualquier función g sobre tales pares puede descomponerse como:

$$g(c, X) = \varphi \left((1 + \epsilon) \cdot f(c) + \sum_{x \in X} f(x) \right)$$

para alguna función φ .



Graph Isomorphic Networks (GIN)

En base a estos resultados, GIN propone aprender f y φ en base al teorema de aproximación universal de MLPs¹. En la práctica, se modela la composición de ambas funciones $f^{(k+1)} \circ \varphi^{(k)}$ con una MLP, dado que la MLP puede aprender la composición de funciones.

En la práctica, en la primera iteración, no necesitamos una MLP antes de la suma de características de entrada, ya que si las features son one hot encodings, por definición estas sumas son inyectivas. Además, epsilon se puede aprender. Luego, la función de update de nodos de GIN se define según:

$$h_v^{(k)} = \text{MLP}^{(k)} \left(\left(1 + \epsilon^{(k)} \right) \cdot h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)} \right)$$

Los embeddings de GIN pueden ser usados directamente para node classification o link prediction.



¹ Hornik, K., Stinchcombe, M., White, H. Multilayer feedforward networks are universal approximators. Neural networks, 2(5):359-366, 1989.

Graph Isomorphic Networks (GIN) – Graph read out

GIN puede generar mediante un graph read out un embedding para el grafo completo. Al igual que el VGAE, el graph read out puede operar a nivel de un subconjunto de los nodos del grafo original.

Los autores de GIN observan que los embeddings de todas las capas de la red aportan información (algo parecido a lo que ocurre con el transformer). Entonces, proponen que el graph read out se obtenga concatenando las representaciones obtenidas en todas las capas:

$$h_G = \text{CONCAT} \left(\text{READOUT} \left(\left\{ h_v^{(k)} \mid v \in G \right\} \right) \mid k = 0, 1, \dots, K \right)$$

En esta ecuación, la función de READOUT puede reemplazarse por la suma, caso en el cual GIN probablemente cumpla con el WL test.

Algunos aspectos prácticos: GIN tiene suficiente capacidad expresiva con MLP de al menos dos capas. Con MLP de una capa no funciona.

GIN muestra que gracias a su inyectividad sobre grafos, es adecuado para tareas de clasificación de grafos. Es decir, es un modelo que permite hacer las tres tareas de GML.

