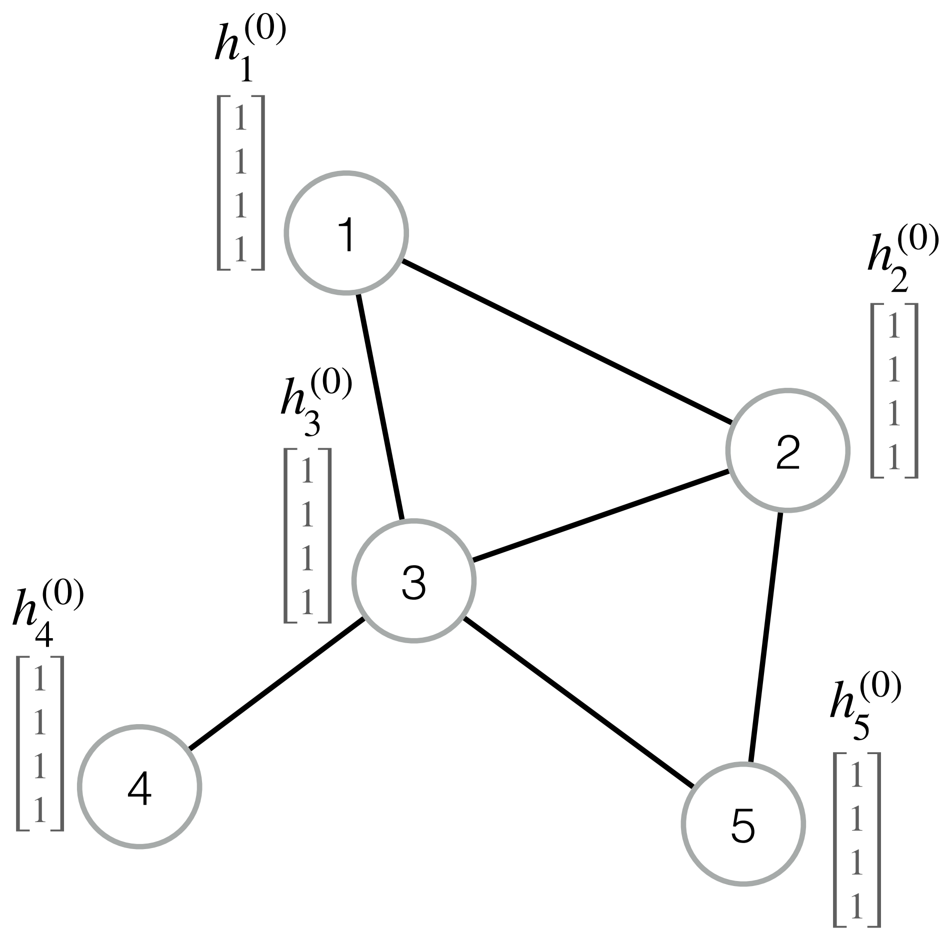
Redes Neuronales de Grafos (GNNs)

(Scarselli et al. 2009, Gilmer et al. 2017)

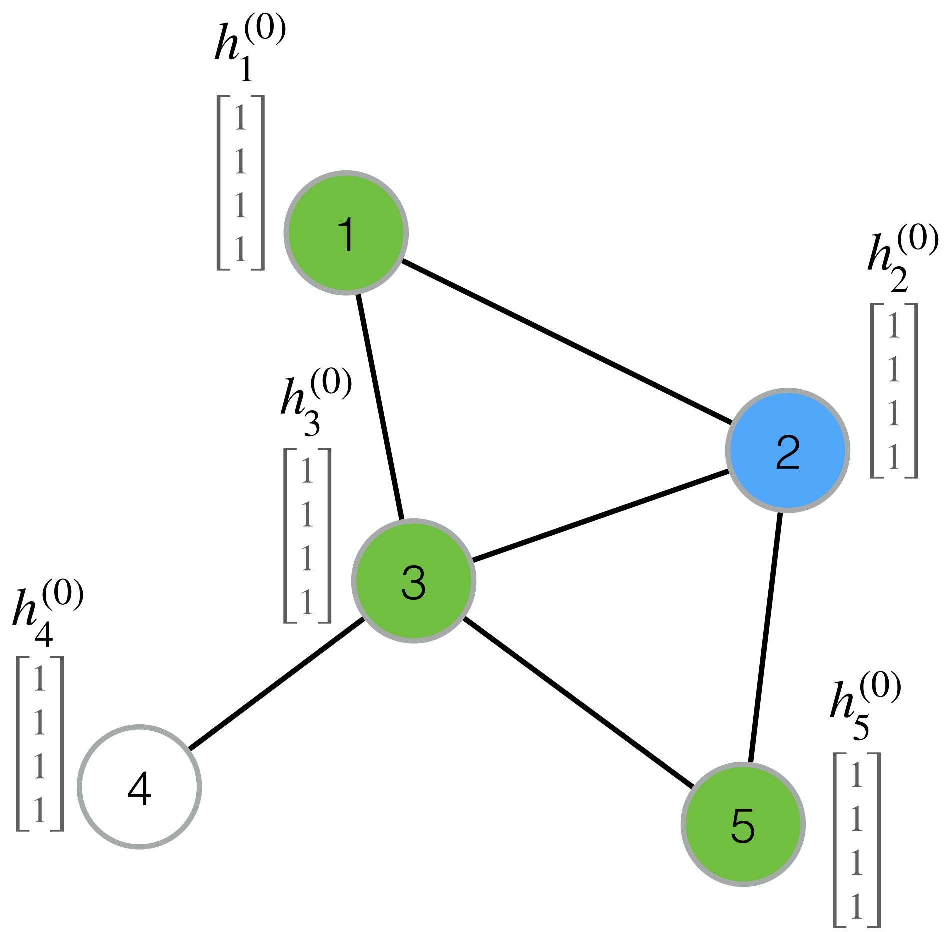


Computación de una GNN:

- Entrada: grafo + vector inicial $h_v^{(0)}$ para cada nodo v
- La GNN tiene T capas
- Cada capa actualiza el vector de cada nodo (usando la información de los vecinos)
- Salida: vector final para cada nodo (última capa T)

Redes Neuronales de Grafos (GNNs)

(Scarselli et al. 2009, Gilmer et al. 2017)



Actualización para nodo 2 en la primera capa:

$$z_2^{(0)} = \mathrm{AGG}(\{\!\{h_1^{(0)}, h_3^{(0)}, h_5^{(0)}\}\!\})$$

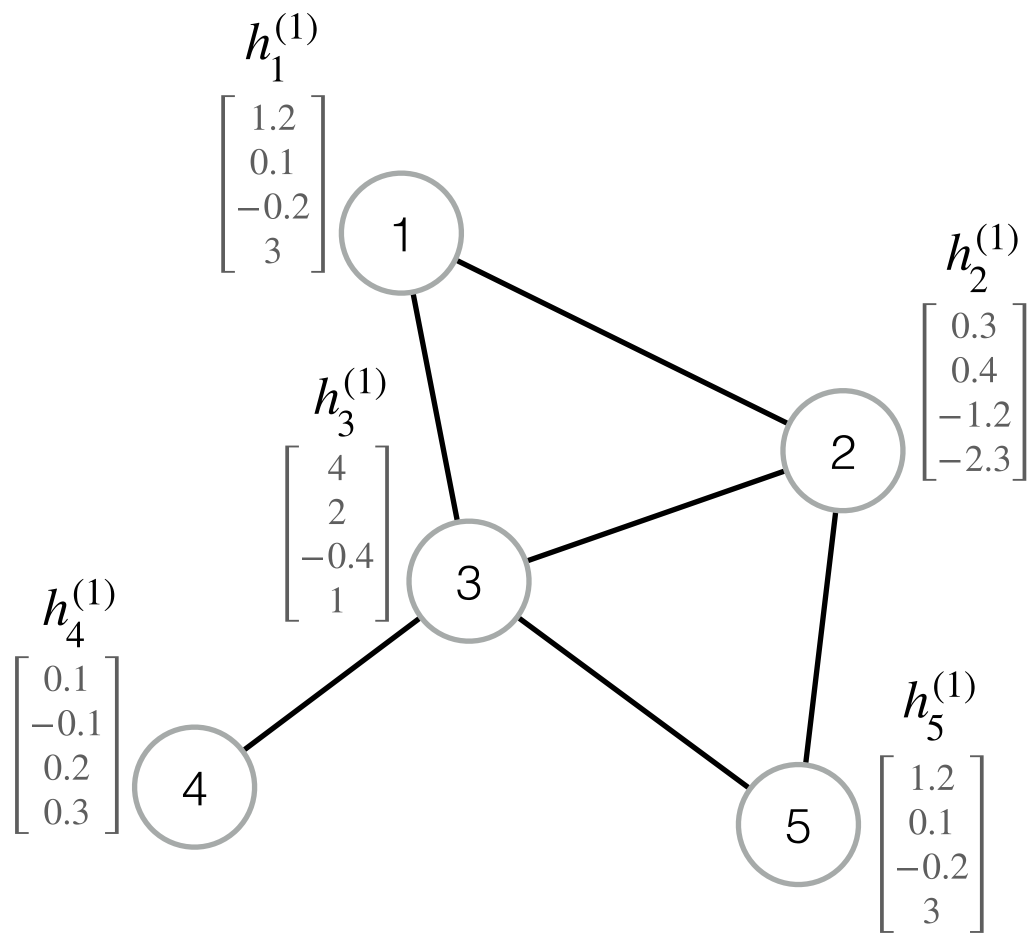
$$h_2^{(1)} = \mathrm{UPD}(h_2^{(0)}, z_2^{(0)})$$

Computación de una GNN:

- Entrada: grafo + vector inicial $h_v^{(0)}$ para cada nodo v
- La GNN tiene T capas
- Cada capa actualiza el vector de cada nodo (usando la información de los vecinos)
- Salida: vector final para cada nodo (última capa T)

Redes Neuronales de Grafos (GNNs)

(Scarselli et al. 2009, Gilmer et al. 2017)

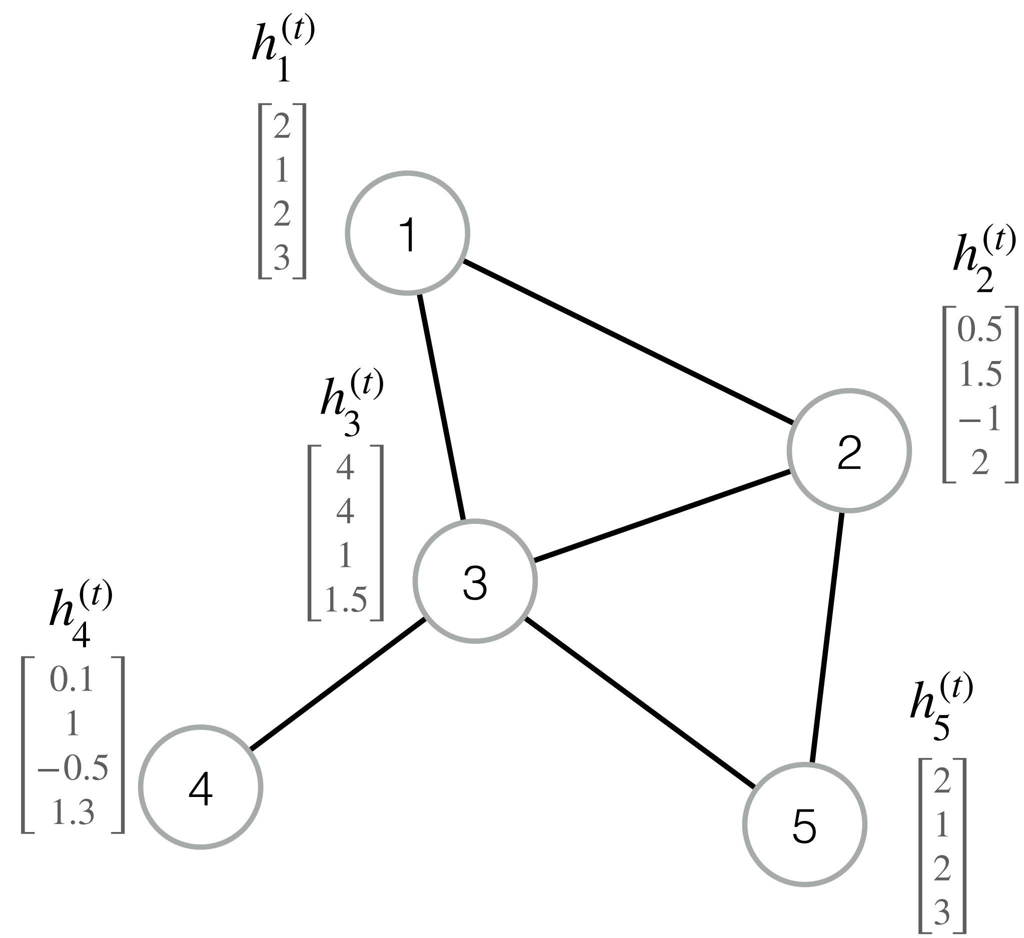


Computación de una GNN:

- Entrada: grafo + vector inicial $h_v^{(0)}$ para cada nodo v
- La GNN tiene T capas
- Cada capa actualiza el vector de cada nodo (usando la información de los vecinos)
- Salida: vector final para cada nodo (última capa T)

Redes Neuronales de Grafos (GNNs)

(Scarselli et al. 2009, Gilmer et al. 2017)



Actualización para nodo v en la capa $t \in \{1, \ldots, T\}$:

$$h_v^{(t)} = \mathrm{UPD}^{(t)}\Big(h_v^{(t-1)}, \mathrm{AGG}^{(t)}(\{\!\{ h_w^{(t-1)} \mid w \in N(v) \}\!\})\Big)$$

funciones con parámetros

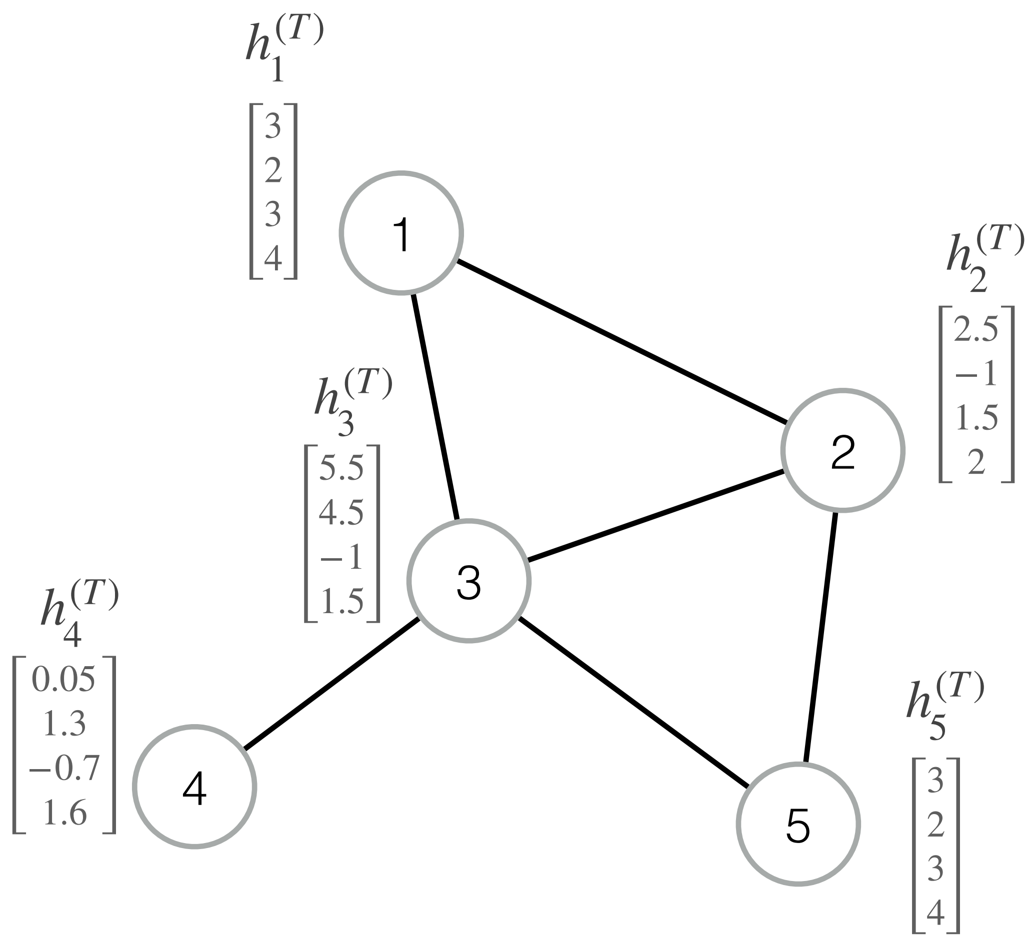
vecinos de v

Computación de una GNN:

- Entrada: grafo + vector inicial $h_v^{(0)}$ para cada nodo v
- La GNN tiene T capas
- Cada capa actualiza el vector de cada nodo (usando la información de los vecinos)
- Salida: vector final para cada nodo (última capa T)

Redes Neuronales de Grafos (GNNs)

(Scarselli et al. 2009, Gilmer et al. 2017)



Actualización para nodo v en la capa $t \in \{1, \ldots, T\}$:

$$h_v^{(t)} = \text{UPD}^{(t)}\Big(h_v^{(t-1)}, \text{AGG}^{(t)}(\{\!\{ h_w^{(t-1)} \mid w \in N(v) \}\!\})\Big)$$

$h_v^{(T)}$ nos da la salida final para nodo v

Computación de una GNN:

- Entrada: grafo + vector inicial $h_v^{(0)}$ para cada nodo v
- La GNN tiene T capas
- Cada capa actualiza el vector de cada nodo (usando la información de los vecinos)
- Salida: vector final para cada nodo (última capa T)

GNNs: arquitecturas

- Existen distintas arquitecturas de este modelo de GNN
  - Varias opciones para las funciones UPD y AGG (suma, promedio, transformaciones lineales, …)

- Ejemplo de arquitectura simple:

  - AGG = suma
  - UPD = transformación lineal + función no lineal

Actualización para nodo v en la capa $t \in \{1,\ldots,T\}$:

$$h_v^{(t)} = \sigma\Big(W_0^{(t)} h_v^{(t-1)} + \sum_{w \in N(v)} W_1^{(t)} h_w^{(t-1)}\Big)$$

Función no lineal
(ej: ReLU)

matrices de parámetros

Poder expresivo de GNNs

- GNNs han mostrado excelentes resultados en los últimos años
  - Datos moleculares/químicos, redes de transporte, redes sociales, …
  - Varias implementaciones (Pytorch, Pytorch Geometric,…)
- Esto ha motivado bastante investigación sobre la teoría de GNNs
- Nos enfocaremos en el poder expresivo de GNNs para distinguir nodos, desde un punto de vista algorítmico
  - Dado un grafo y una GNN, ¿Qué nodos es capaz de distinguir la GNN?
  - ¿Es posible cuantificar esto de una manera precisa?
  - Relacionado con clasificación de nodos

El test Weisfeiler-Leman (1-WL)

- 1-WL: Dado un grafo $G = (V, E)$
  - Comenzamos con un coloreo para los nodos $(c_v^{(0)})_{v \in V}$
  - En la iteración $t \geq 1$, actualizamos el color de cada nodo según los colores de sus vecinos:

$$c_v^{(t)} = \text{Relabel}\Big((c_v^{(t-1)}, \{\!\{ c_w^{(t-1)} \mid w \in N(v) \}\!\})\Big)$$

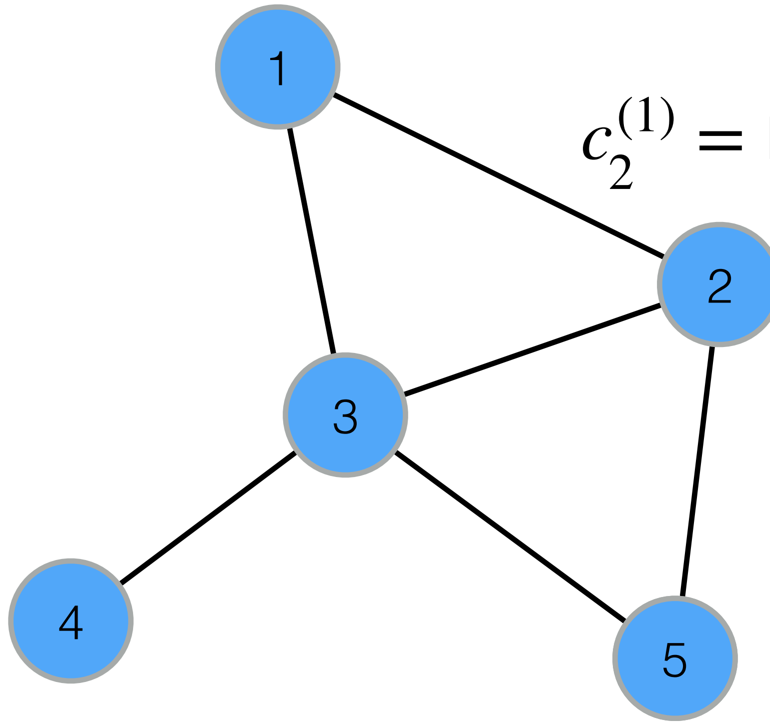
  - Podemos aplicar el test una cantidad fija de iteraciones o hasta que el coloreo se estabilice

(Es decir, hasta que la partición obtenida no cambie)

1-WL: ejemplo

$c_1^{(1)} = \text{Relabel}\big((\bullet, \{\!\{\bullet, \bullet\}\!\})\big) = \bullet$

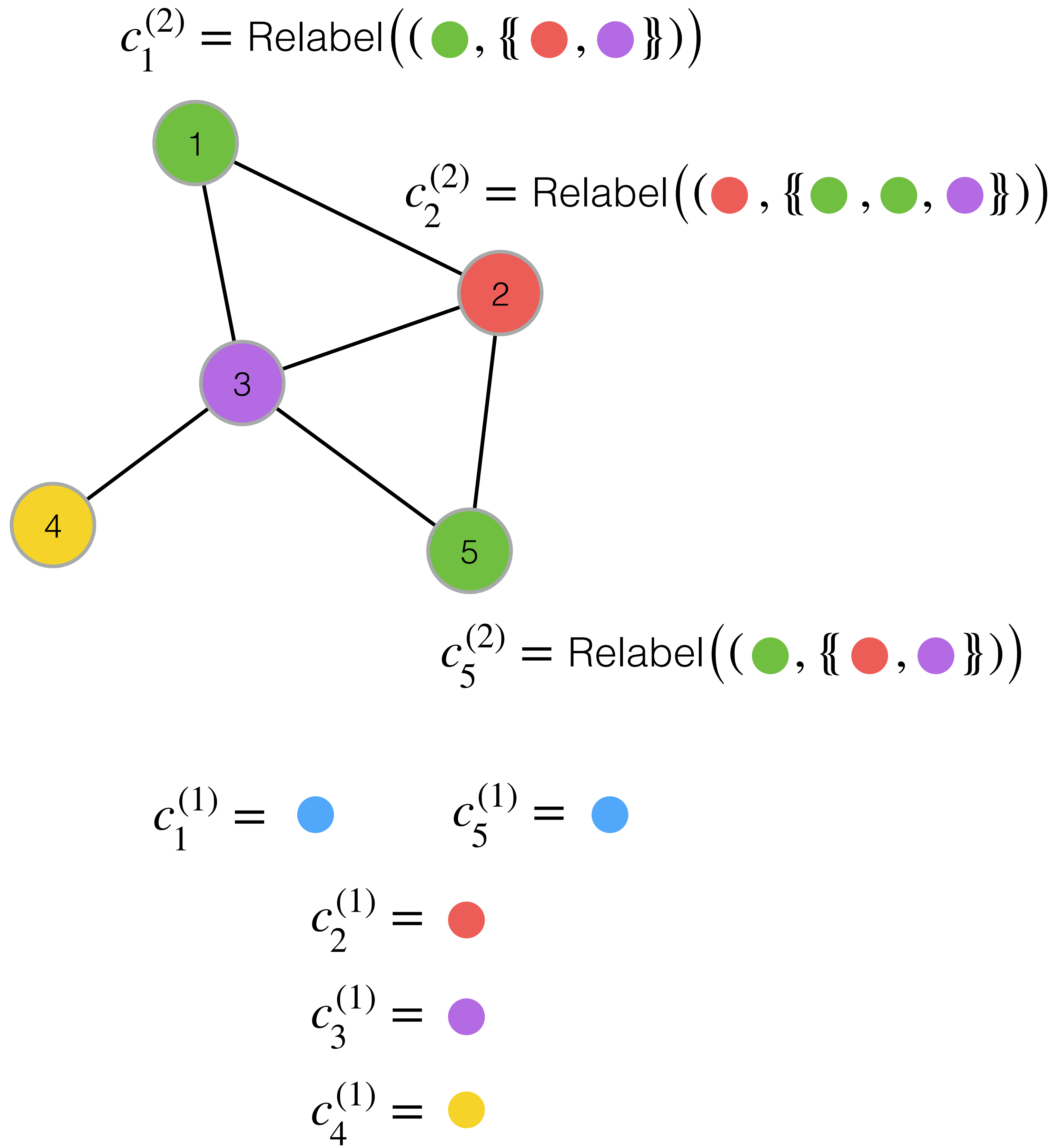
$c_2^{(1)} = \text{Relabel}\big((\bullet, \{\!\{\bullet, \bullet, \bullet\}\!\})\big) = \bullet$



$c_5^{(1)} = \text{Relabel}\big((\bullet, \{\!\{\bullet, \bullet\}\!\})\big) = \bullet$

$c_v^{(0)} = \bullet$ para todo nodo $v \in V$

1-WL: ejemplo



$c_1^{(2)} =$ Relabel
1
$c_2^{(2)} =$ Relabel
2
3
4
5
$c_5^{(2)} =$ Relabel
$c_1^{(1)} =$
$c_5^{(1)} =$
$c_2^{(1)} =$
$c_3^{(1)} =$
$c_4^{(1)} =$

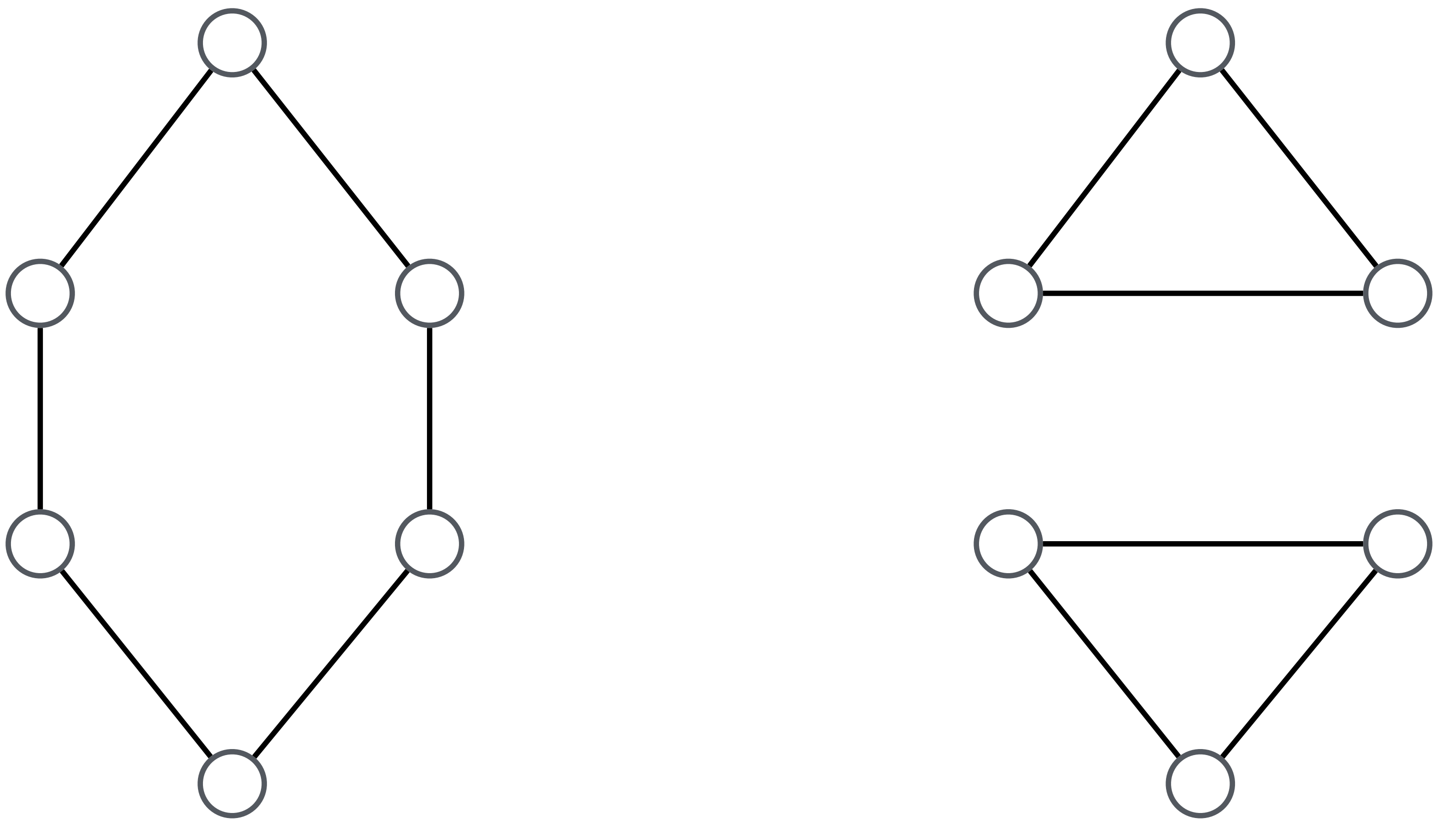
1-WL e isomorfismo

- El test 1-WL es una heurística clásica para isomorfismo
  - Tenemos dos grafos G y H, y queremos verificar si son isomorfos.
  - Aplicamos 1-WL por separado en G y H:
    - En ambos grafos el coloreo inicial le asigna el mismo color a todos los nodos
  - Aplicamos 1-WL hasta que los coloreos se estabilicen
  - Si los histogramas de los colores es el mismo para G y H, entonces el test **acepta**; sino, **rechaza**.

1-WL es una heurística (no es siempre correcto):

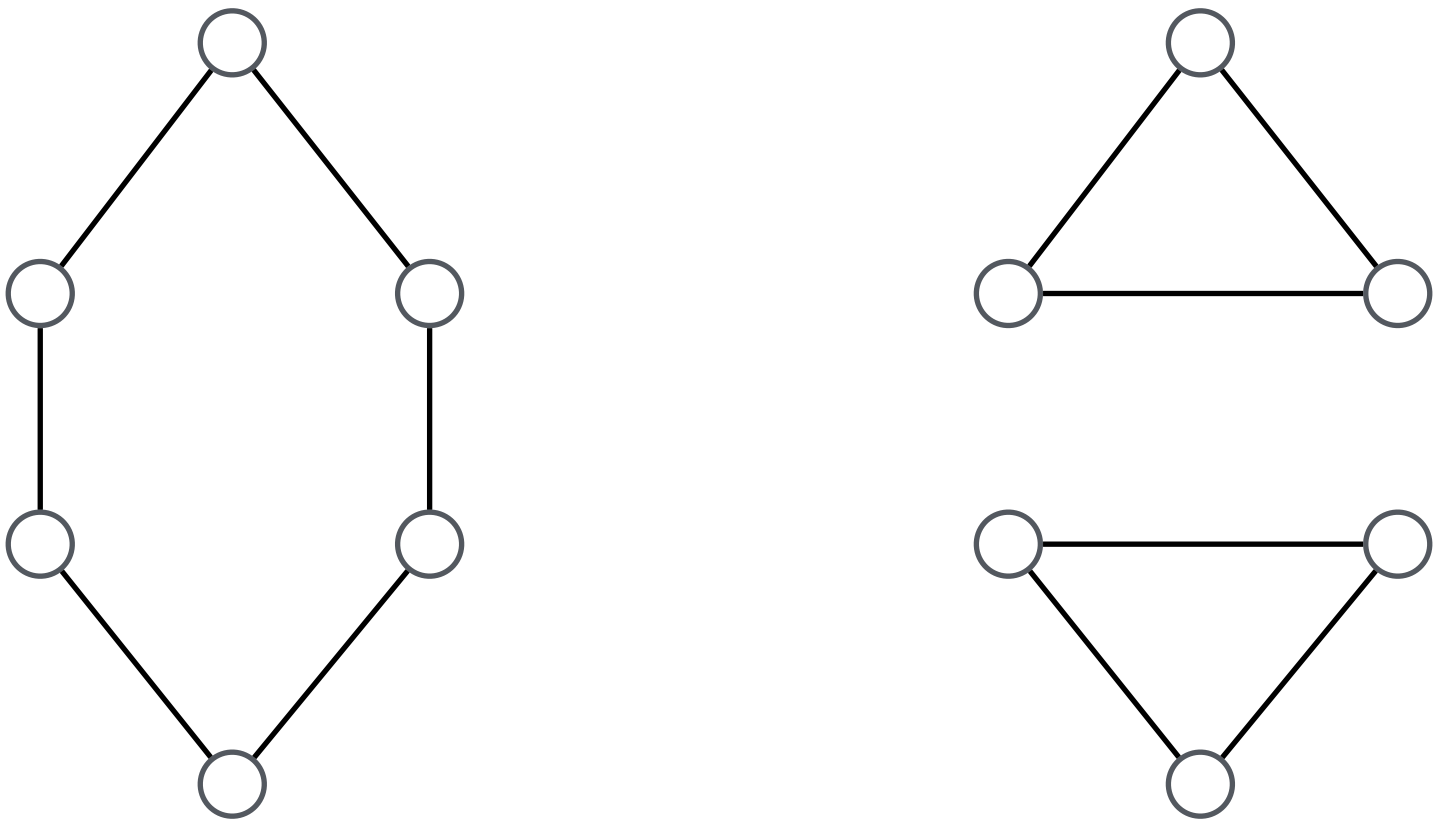
- Si G y H son isomorfos, entonces el test acepta.
- Si G y H no son isomorfos, el test podría aceptar!

1-WL e isomorfismo



Acá 1-WL acepta, pero los grafos no son isomorfos

1-WL e isomorfismo



Acá 1-WL acepta, pero los grafos no son isomorfos

1-WL y GNNs

- Hay una fuerte conexión entre GNNs y 1-WL:
  - El poder expresivo de las GNNs para distinguir nodos **coincide** con el de 1-WL

Cota superior: Las GNNs no son más poderosas que 1-WL

Teorema (Morris et al. 2019, Xu et al. 2019):

Sea $G = (V, E)$ un grafo y $T \geq 1$ un entero positivo.

Sean $(c_v^{(0)})_{v \in V}$ y $(h_v^{(0)})_{v \in V}$ coloreos y features iniciales equivalentes (generan la misma partición).

Entonces, toda GNN con T capas cumple:

$$c_u^{(t)} = c_v^{(t)} \implies h_u^{(t)} = h_v^{(t)}$$

para todo $0 \leq t \leq T$, y para todo par de nodos $u, v \in V$

1-WL y GNNs

- Hay una fuerte conexión entre GNNs y 1-WL:
  - El poder expresivo de las GNNs para distinguir nodos **coincide** con el de 1-WL

Cota inferior: Las GNNs pueden ser igual de poderosas que 1-WL

Teorema (Morris et al. 2019):

Sea $G = (V, E)$ un grafo y $T \geq 1$ un entero positivo.

Sea $(c_v^{(0)})_{v \in V}$ un coloreo inicial para G.

Entonces, existen features iniciales $(h_v^{(0)})_{v \in V}$ equivalentes a $(c_v^{(0)})_{v \in V}$
y una GNN con T capas tal que:

$$c_u^{(t)} = c_v^{(t)} \iff h_u^{(t)} = h_v^{(t)}$$

para todo $0 \leq t \leq T$, y para todo par de nodos $u, v \in V$

Observación: Se puede escoger una GNN con arquitectura simple

Poder expresivo de GNNs

- El resultado anterior nos dice que las GNNs tienen los mismos límites que 1-WL (para distinguir nodos)

- Se conocen varias extensiones de 1-WL que son mas poderosas.

- Estas extensiones han inspirado nuevas arquitecturas de GNNs con un mayor poder expresivo
  (Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. Morris et al. 2019)

El test k-WL

- El test k-WL es más expresivo que 1-WL, en el sentido que puede colorear tuplas de k nodos

- Notación:
  - Dado un grafo $G = (V, E)$ y una tupla $s \in V^k$ definimos su j-ésimo vecindario:

$$N_j(s) = \{(s_1, \ldots, s_{j-1}, r, s_{j+1}, \ldots, s_k) \mid r \in V\}$$

- k-WL: Dado un grafo $G = (V, E)$
  - En la iteración 0, cada tupla $s \in V^k$ se colorea con su "isomorphism type"
  - En la iteración $t \geq 1$, actualizamos el color de cada tupla $s \in V^k$ según:

$$c_s^{(t)} = \text{Relabel}\Big(\big(c_s^{(t-1)}, C_1^{(t)}(s), \ldots, C_k^{(t)}(s)\big)\Big)$$

$$C_j^{(t)}(s) = \text{Relabel}\Big(\{\!\{ c_{s'}^{(t-1)} \mid s' \in N_j(s) \}\!\}\Big)$$

- Existen varias variantes de k-WL y arquitecturas de GNNs asociadas (Morris et al. 2019)