The Logical Expressiveness of GNNs

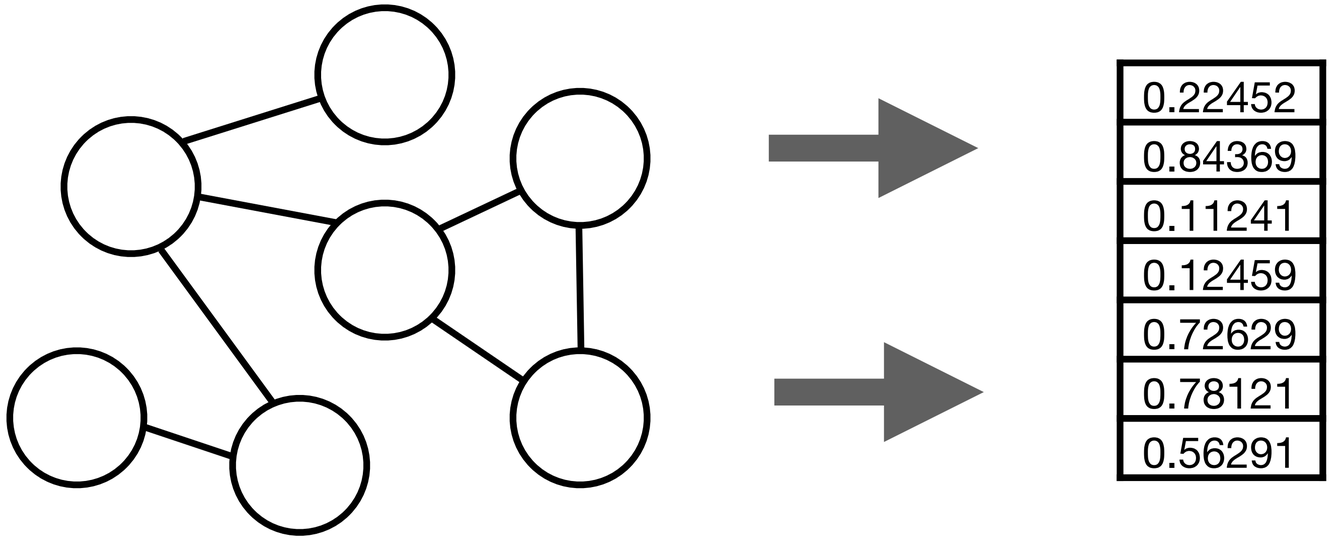
Basado en

The logical expressiveness of graph neural networks
Pablo Barceló, Egor V Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, Juan Pablo Silva
ICLR 2020

Expressiveness and approximation properties of graph neural networks
Floris Geerts, Juan L Reutter
ICLR 2022

Graph Neural Networks

GNNs



aprenden *graph embeddings*

Capa MPNN (Message Passing Neural Network)

Nuevos embeddings $emb_G(v)$
De embeddings antiguos $emb'_G(v)$

1. Agregar la información de mis vecinos
2. Hacer Update del agregado con lo que tenía antes
3. Pasar todo por una función de activación no lineal

Capa MPNN (Message Passing Neural Network)

Nuevos embeddings $emb_G(v)$
De embeddings antiguos $emb'_G(v)$

$$emb_G(v) := \sigma\Big(\text{Update}\big(emb'_G(v)\,, \text{Aggregate}\big(\{\{\, emb'_G(w) \mid w \in N_G(v) \,\}\}\big)\big)\Big)$$

Indice

1. Poder de separación - repaso clase pasada
2. Qué hacen realmente las GNNs? Versión lógica
3. Qué hacen realmente las GNNs? Version funciones aproximadas

*** si queda tiempo (slides en inglés perdón)

4. Beyond simple GNNs
5. How to study them

Separation Power

Separation Power

Relación de equivalencia dada por grafos/nodos para los que una GNN computa el mismo embedding

Versión grafo

$$(G, H) \in \rho(GNN) \Leftrightarrow emb_G = emb_H$$

Versión nodos

$$(G, v, w) \in \rho(GNN) \Leftrightarrow emb_G(v) = emb_G(w)$$

Separation Power

Relación de equivalencia dada por grafos/nodos para los que una GNN computa el mismo embedding

Versión grafo

$$(G, H) \in \rho(GNN) \Leftrightarrow emb_G = emb_H$$

Versión nodos

$$(G, v, w) \in \rho(GNN) \Leftrightarrow emb_G(v) = emb_G(w)$$

Mas pequeña la relación ⟶ **Mas separation power**

Separation Power

Relación de equivalencia dada por grafos/nodos para los que una GNN computa el mismo embedding

Versión grafo

$$(G, H) \in \rho(GNN) \Leftrightarrow emb_G = emb_H$$

Mas pequeña la relación ⟶ **Mas separation power**

$\rho(GNN_1) \subseteq \rho(GNN_2)$: GNN_1 distingue más grafos que GNN_2

Separation Power

Relación de equivalencia dada por grafos/nodos para los que una GNN computa el mismo embedding

Versión nodos

$$(G, v, w) \in \rho(GNN) \Leftrightarrow emb_G(v) = emb_G(w)$$

Mas pequeña la relación ⟶ **Mas separation power**

$\rho(GNN_1) \subseteq \rho(GNN_2)$:

hay grafos donde GNN_1 distingue nodos que no distingue GNN_2

Separation Power

Relación de equivalencia dada por grafos/nodos para los que una GNN computa el mismo embedding

Nos interesa el separation power de architectural (clases) de GNNs.
Que arquitecturas distinguen más grafos/nodos.

Separation Power

Relación de equivalencia dada por grafos/nodos para los que una GNN computa el mismo embedding. Para clase $\mathbf{C}$ de GNNs:

Versión grafo

$$(G, H) \in \rho(\mathbf{C}) \Leftrightarrow emb_G = emb_H$$

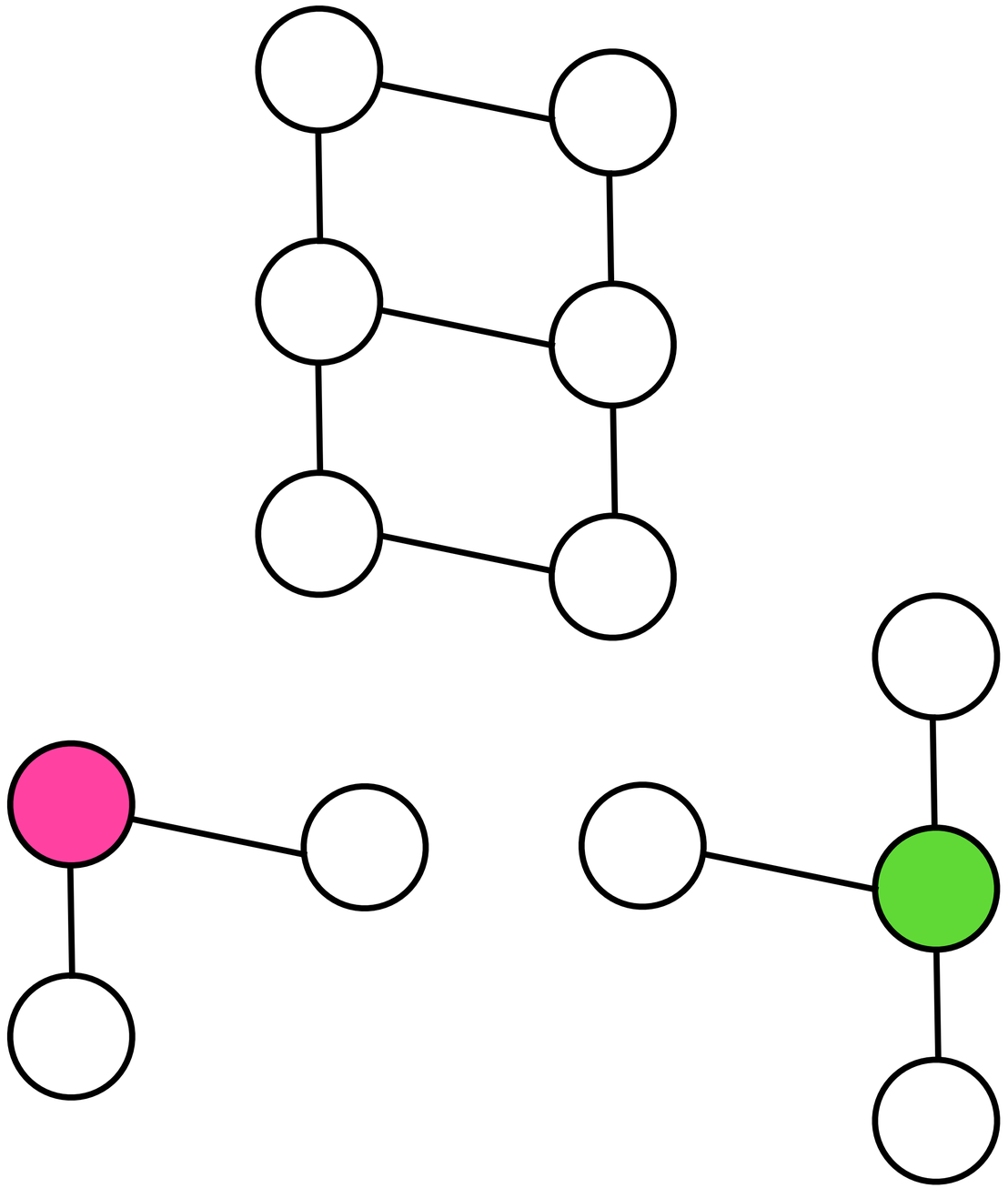
Versión nodos

$$(G, v, w) \in \rho(\mathbf{C}) \Leftrightarrow emb_G(v) = emb_G(w)$$

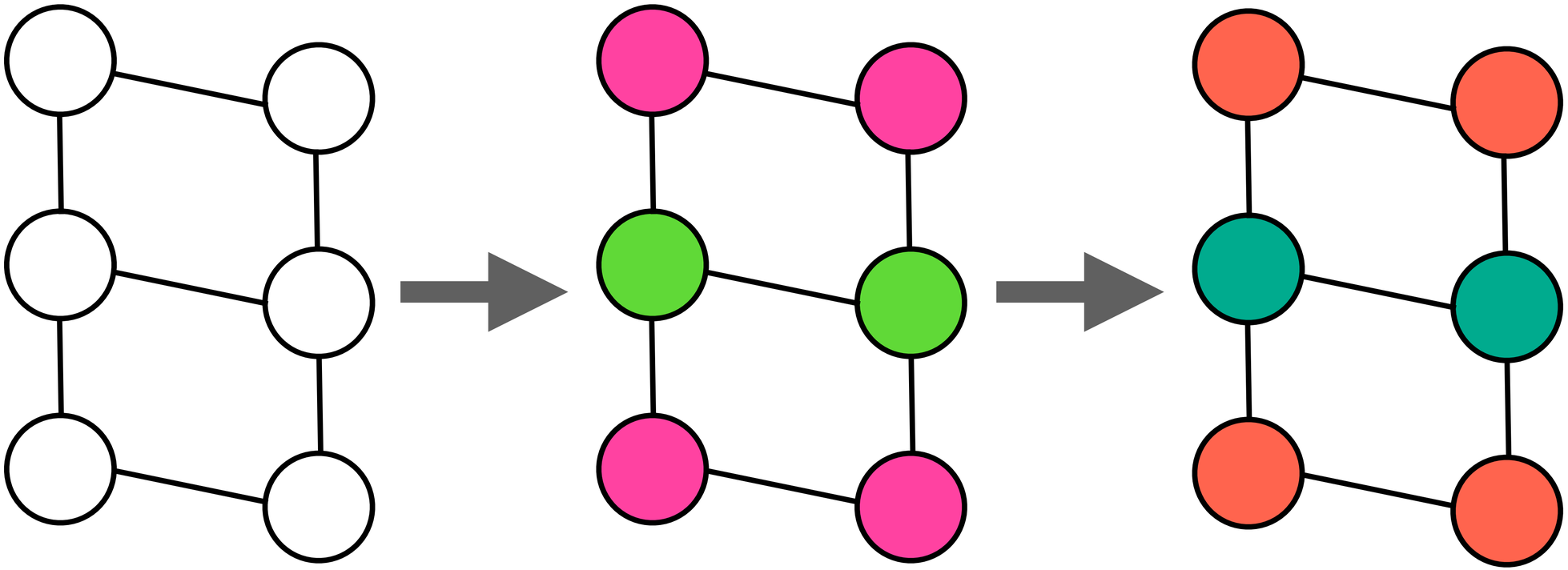
Mas pequeña la relación ⟶ **Mas separation power**

Weisfeiler-Lehman (Colour refinement)

- Color inicial basado en features
- Actualizamos según multiset de vecinos
- Paramos cuando no separamos un nuevo nodo.

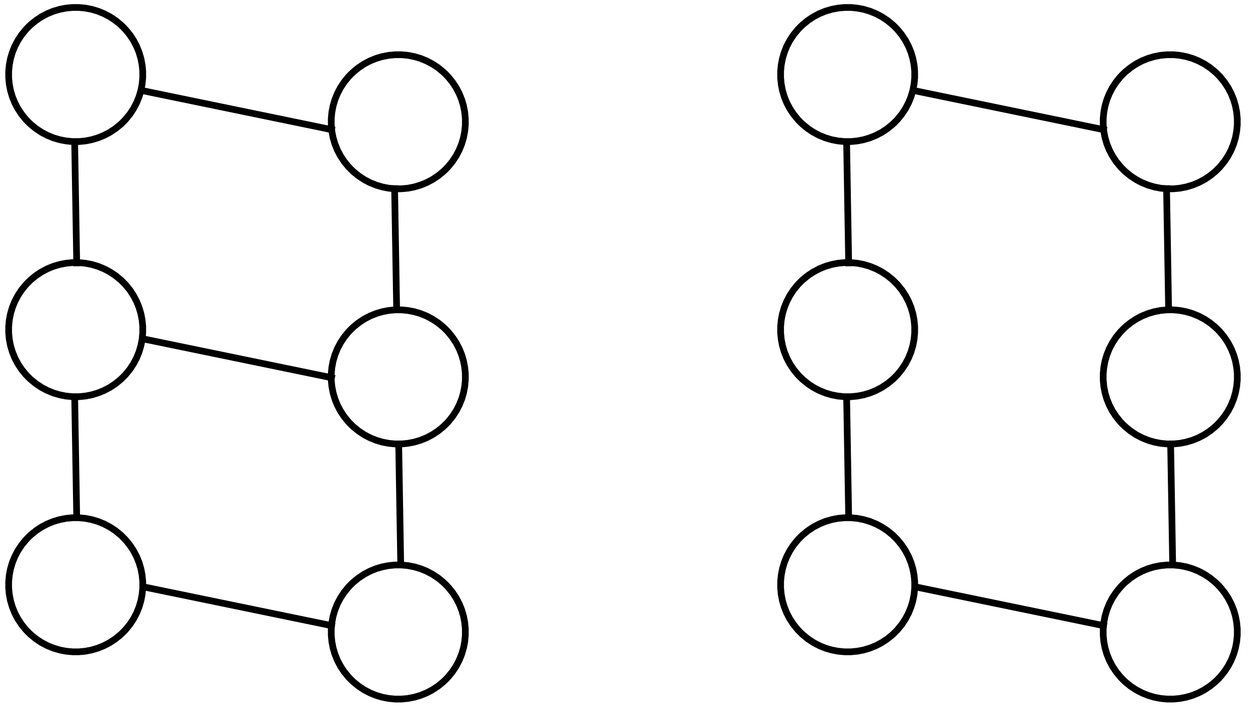


Weisfeiler-Lehman (Colour refinement)



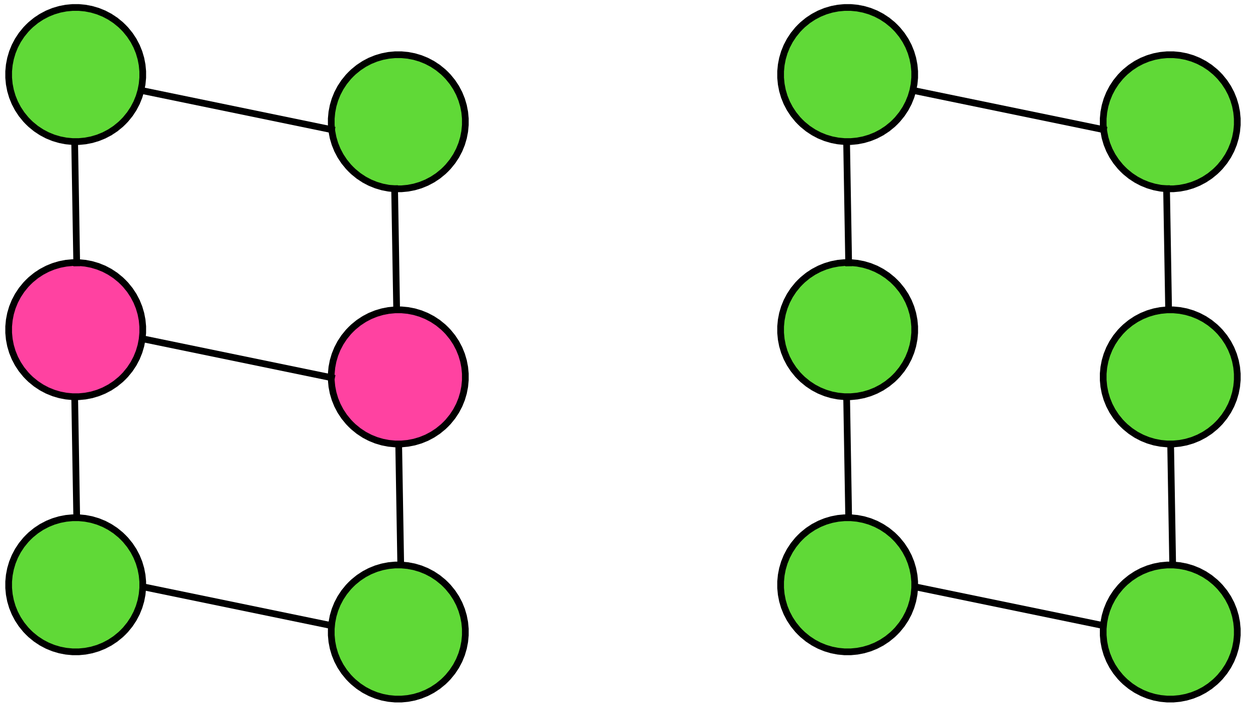
Aqui paramos

Colour refinement ~ graph embeddings



Los colores de los vértices los podemos entender como embeddings

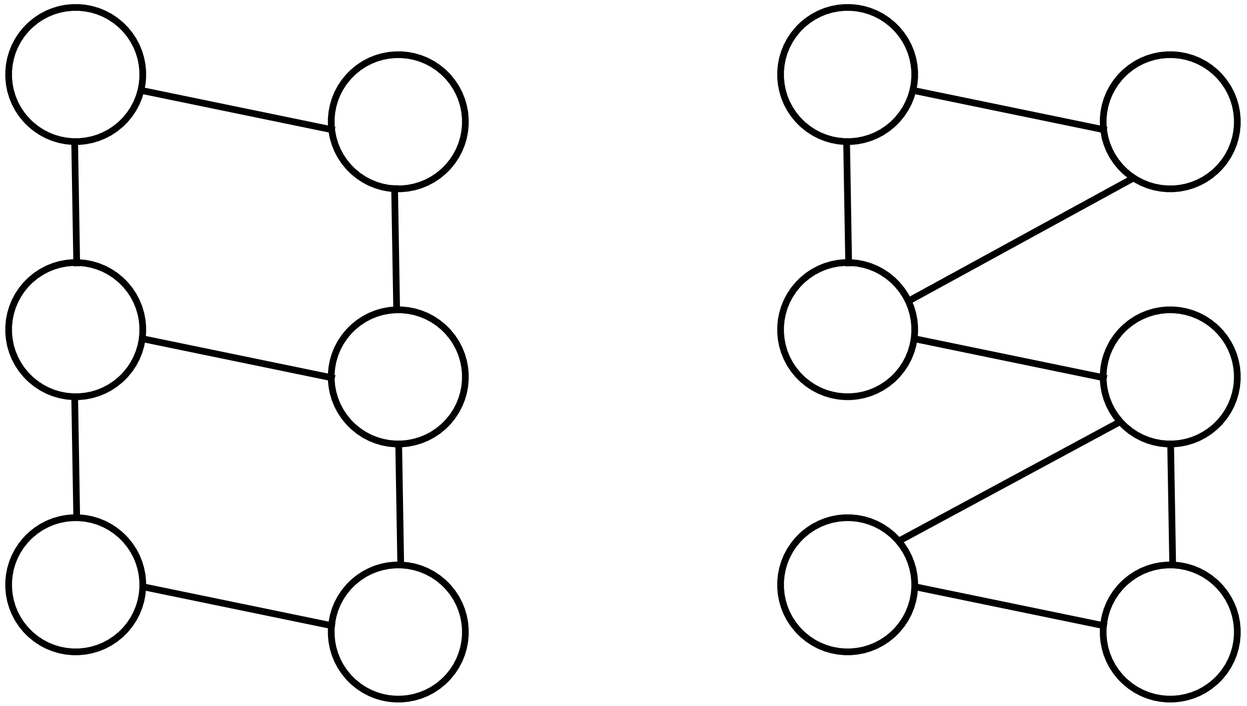
Colour refinement ~ graph embeddings



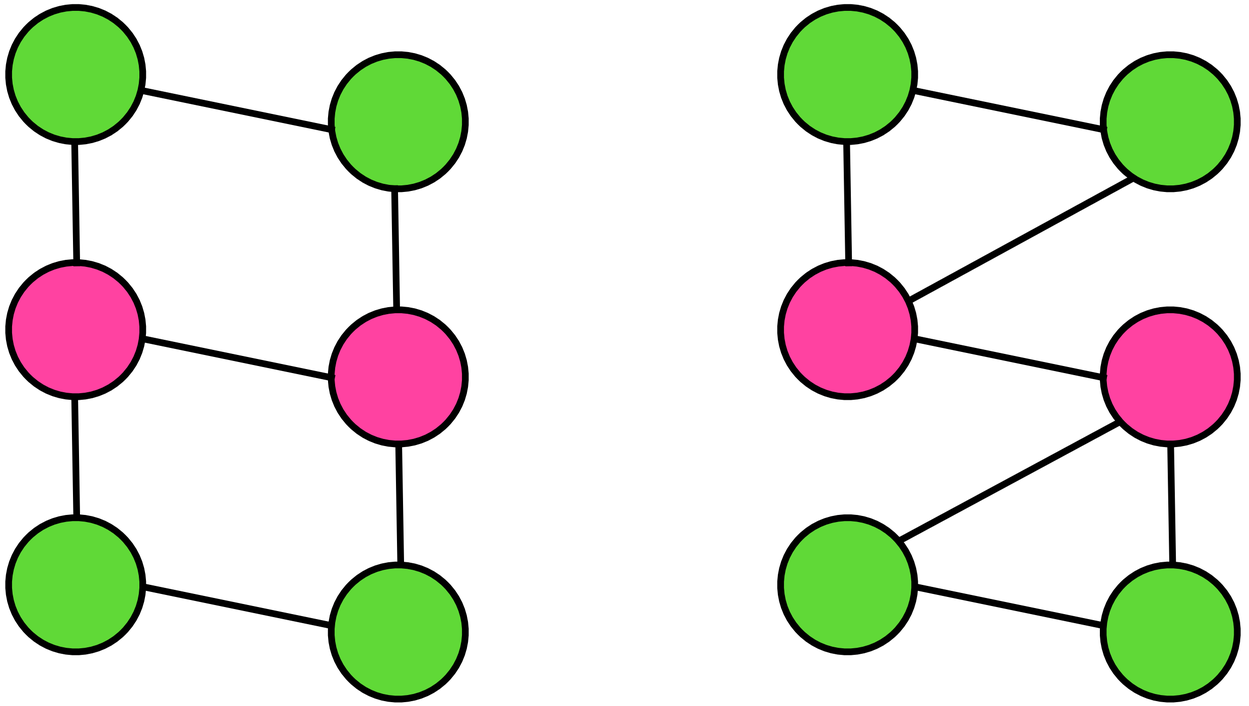
Grafos diferentes:

Multiset de colores de nodos es diferente

Colour refinement ~ graph embeddings



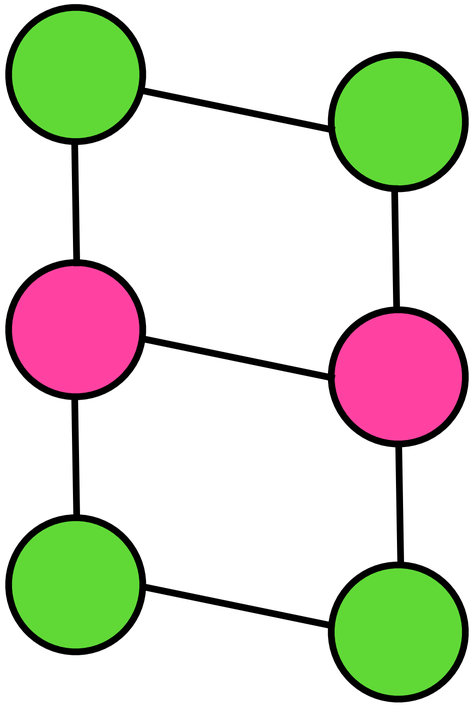
Colour refinement ~ graph embeddings



El mismo embedding de grafo

Estos grafos no son separados

Colour refinement ~ node embeddings



Los nodos del mismo color tampoco son separados

Poder de separación de MPNNs

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

El poder de separación de clases de MPNNs es superconjunto de:

- **Colour refinement, para node embeddings**
- **1-dimensional WL algorithm, para embeddings de grafos**

(La diferencia entre estos dos es que 1-WL agrega sobre todos los nodos, no los vecinos)

Poder de separación de MPNNs

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

El poder de separación de clases de MPNNs es superconjunto de:

- **Colour refinement, para node embeddings**
- **1-dimensional WL algorithm, para embeddings de grafos**

Toma una (clase de) MPNN cualquiera M

Luego,

$\rho(CR) \subseteq \rho(M)$ (si ρ es sobre nodos)

$\rho(1\text{-}WL) \subseteq \rho(M)$ (si ρ es sobre grafos)

Poder de separación de MPNNs

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

El poder de separación de clases de MPNNs es superconjunto de:

- **Colour refinement, para node embeddings**
- **1-dimensional WL algorithm, para embeddings de grafos**

Toma una (clase de) MPNN cualquiera M

Luego,

$\rho(CR) \subseteq \rho(M)$ (si ρ es sobre nodos)

$\rho(1\text{-}WL) \subseteq \rho(M)$ (si ρ es sobre grafos)

Poder de separación de MPNNs

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

El poder de separación de clases de MPNNs es superconjunto de:

- **Colour refinement, para node embeddings**
- **1-dimensional WL algorithm, para embeddings de grafos**

Si Colour Refinement no distingue que dos nodos son distintos
Ninguna MPNN los distingue tampoco

Poder de separación de MPNNs

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

El poder de separación de clases de MPNNs es superconjunto de:

- **Colour refinement, para node embeddings**
- **1-dimensional WL algorithm, para embeddings de grafos**

Si 1-WL no distingue que dos grafos son distintos
Ninguna MPNN los distingue tampoco

Poder de separación de MPNNs

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

El poder de separación de clases de MPNNs es superconjunto de:

- **Colour refinement, para node embeddings**
- **1-dimensional WL algorithm, para embeddings de grafos**

Finalmente, hay arquitecturas que aproximan los resultados de 1-WL y Colour Refinement sobre, para cualquier precisión

Preguntas al vacío

Para grafos, CR y 1-WL es lo mismo
Para nodos, no

Que pasa con una MPNN cualquiera, ¿qué está computando?
Algo que está acotado por CR/1-WL. ¿Pero qué?

Indice

1. Poder de separación - repaso clase pasada
2. Qué hacen realmente las GNNs? Versión lógica
3. Qué hacen realmente las GNNs? Version funciones aproximadas

*** si queda tiempo (slides en inglés perdón)

4. Beyond simple GNNs
5. How to study them

La lógica FOC2

FO2: Lógica de primer orden con solo 2 variables

$$Red(x) \wedge \exists y\Big(Edge(x,y) \wedge \exists x[Edge(y,x) \wedge Blue(x)]\Big)$$

$$Red(x) \wedge \exists y\Big(Edge(x,y) \wedge \exists x[Edge(y,x) \wedge \exists y[Edge(x,y) \wedge Blue(y)]]\Big)$$

Captura bien idea de caminos
No consigue capturar vecindarios

La lógica FOC2

FOC2: Lógica de primer orden con solo 2 variables, pero con $\exists^{\geq N}$

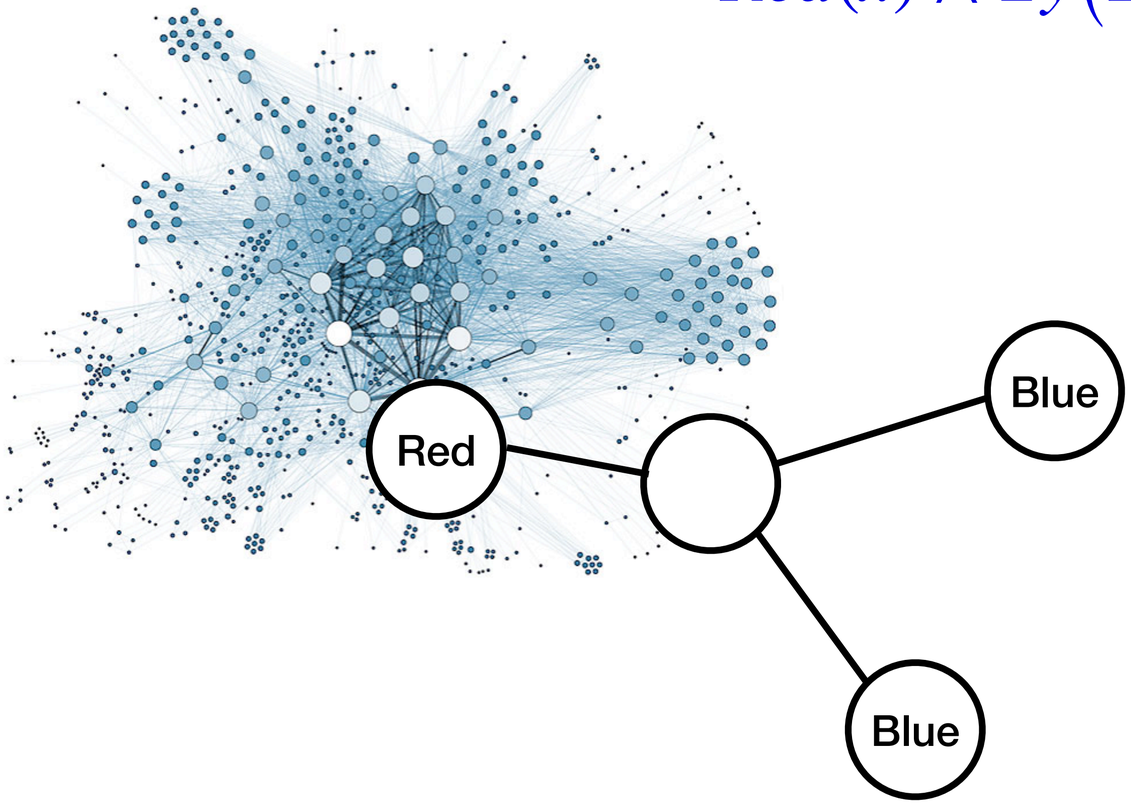
$$Red(x) \wedge \exists y\Big(Edge(x,y) \wedge \exists x[Edge(y,x) \wedge Blue(x)]\Big)$$

$$Red(x) \wedge \exists y\Big(Edge(x,y) \wedge \exists^{\geq 2} x[Edge(y,x) \wedge Blue(x)]\Big)$$

$$Red(x) \wedge \exists y\Big(\neg Edge(x,y) \wedge \exists^{\geq 2} x[Edge(y,x) \wedge Blue(x)]\Big)$$

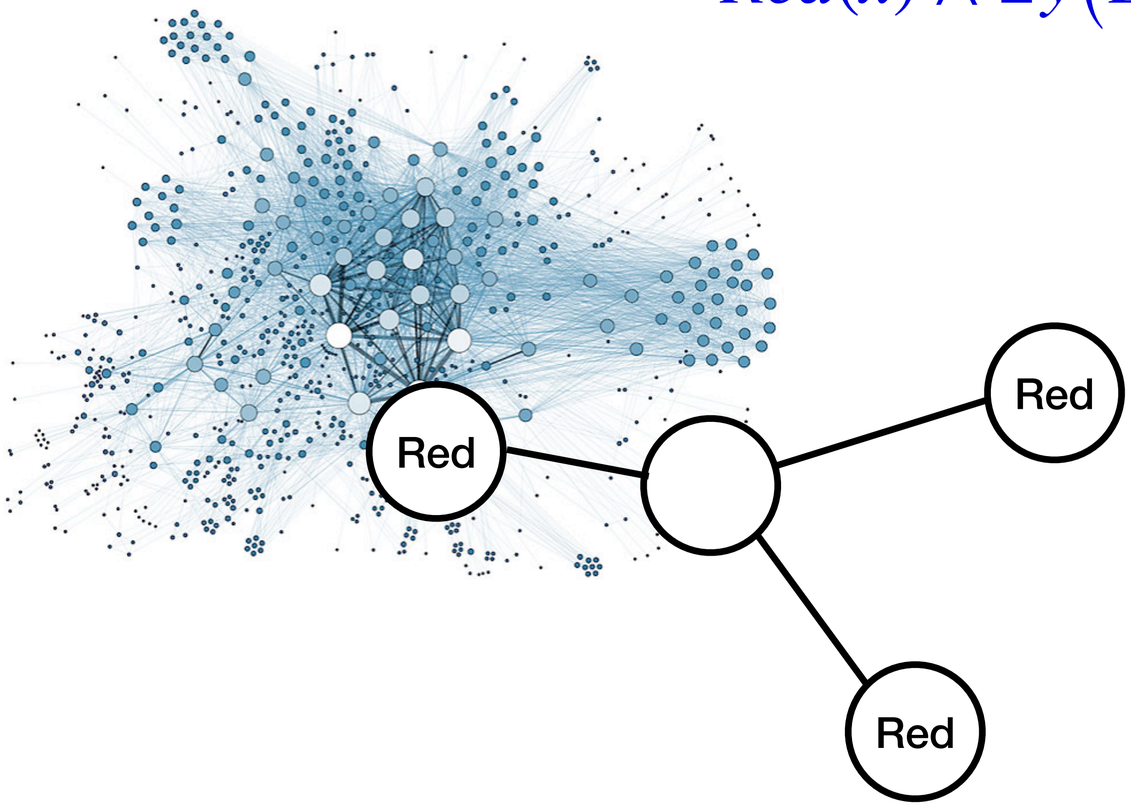
FOC2, tree unravellings

$$Red(x) \wedge \exists y\Big(Edge(x,y) \wedge \exists^{\geq 2} x[Edge(y,x) \wedge Blue(x)]\Big)$$



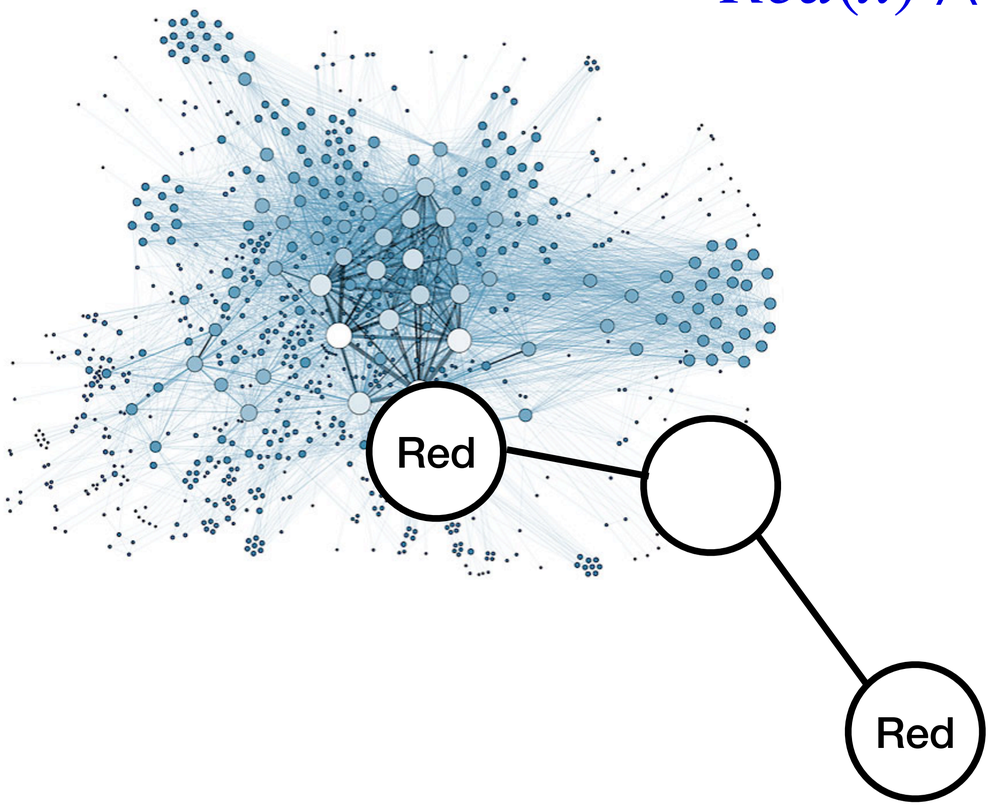
FOC2, tree unravellings

$$Red(x) \wedge \exists y\Big(Edge(x,y) \wedge \exists^{\geq 2} x[Edge(y,x) \wedge Red(x)]\Big)$$



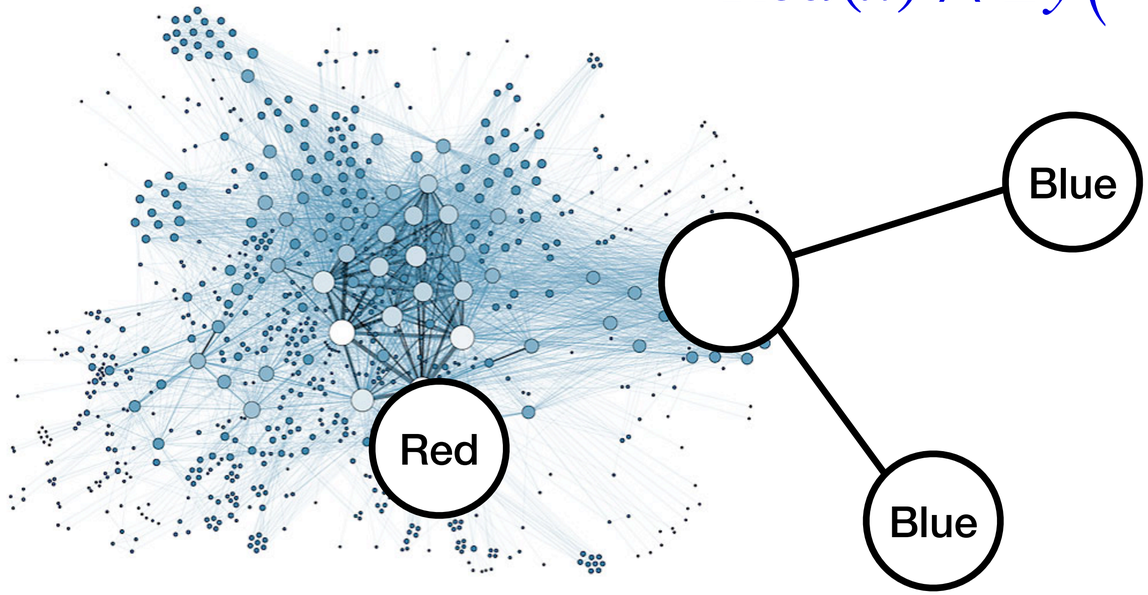
FOC2, tree unravellings

$$Red(x) \wedge \exists y\Big(Edge(x,y) \wedge \exists^{\geq 2} x[Edge(y,x) \wedge Red(x)]\Big)$$



FOC2, tree unravellings

$$Red(x) \wedge \exists y\Big(\neg Edge(x,y) \wedge \exists^{\geq 2} x[Edge(y,x) \wedge Blue(x)]\Big)$$



La lógica FOC2

FOC2: Lógica de primer orden con solo 2 variables, pero con $\exists^{\geq N}$

Teorema (Cai et al.):

Los siguientes son equivalentes en cualquier grafo:

- **Dos nodos tienen el mismo Colour Refinement**
- **Dos nodos son equivalentes con respecto a FOC2**

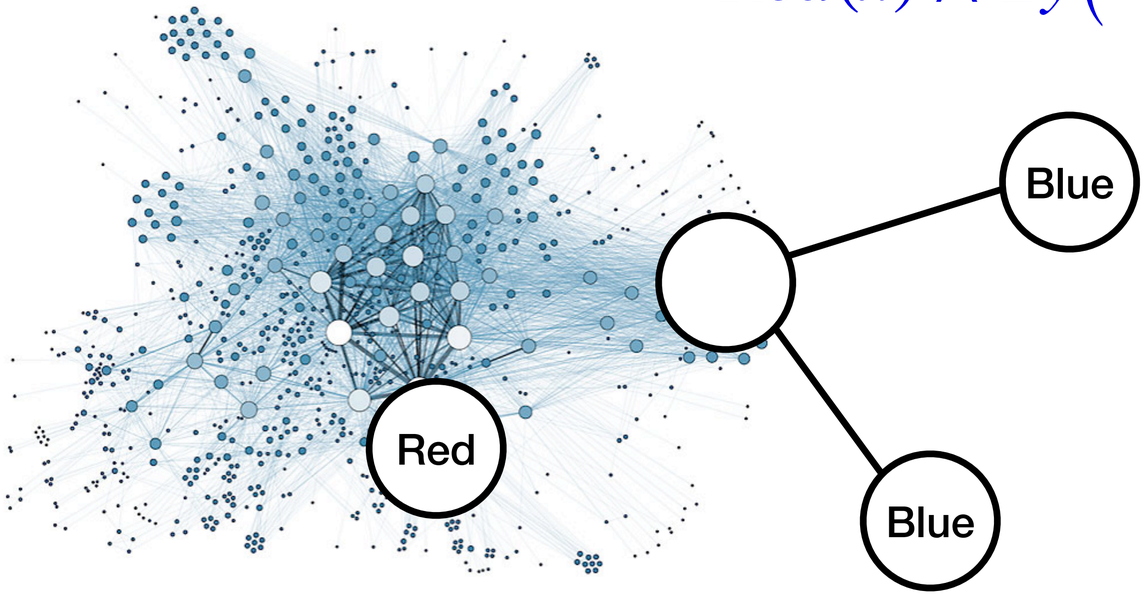
Preguntas al vacío

Para grafos, CR y 1-WL es lo mismo
Para nodos, no

Que pasa con una MPNN cualquiera, ¿qué está computando?
Algo que está acotado por CR/1-WL. ¿Pero qué?

Ejemplo FOC2 difícil

$$Red(x) \wedge \exists y\Big(\neg Edge(x,y) \wedge \exists^{\geq 2} x[Edge(y,x) \wedge Blue(x)]\Big)$$



Esta fórmula no la distingue ninguna MPNN!

Podemos ver por qué?

Poder de MPNN como logical classifiers

Tomamos todas las funciones de grafos a {0,1},

Tomamos todas las funciones de grafos y un nodo a {0,1}

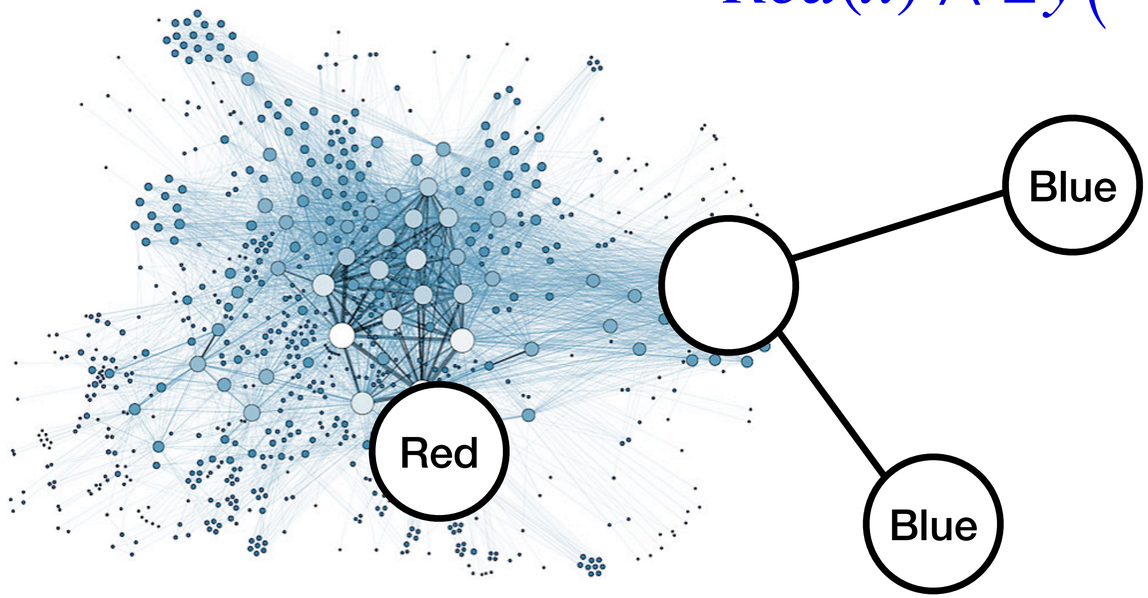
Llamemosle funciones lógicas. Cuáles de estas se capturan por MPNNs?

Teorema (Barceló et al, 2020):

Hay una función lógica de nodos que se escribe en FOC2,
Pero no puede ser capturada por una MPNN

Ejemplo FOC2 difícil

$$Red(x) \wedge \exists y\Big(\neg Edge(x,y) \wedge \exists^{\geq 2} x[Edge(y,x) \wedge Blue(x)]\Big)$$

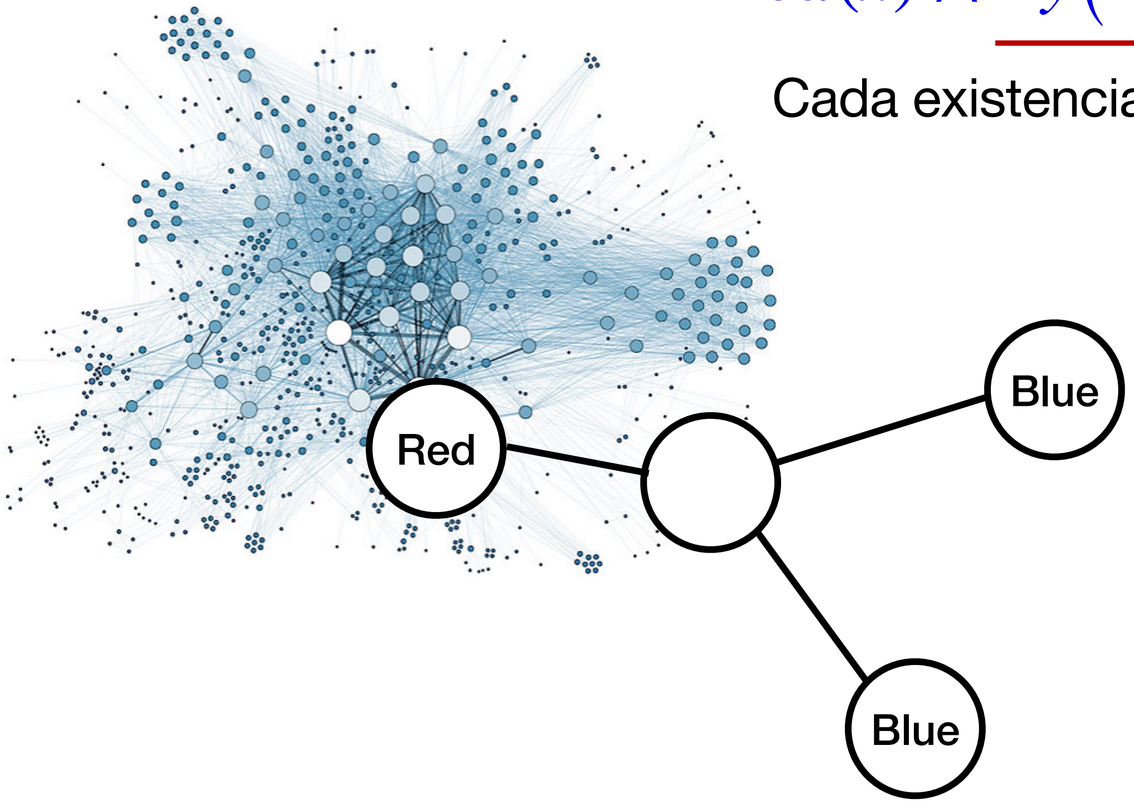


El problema es que FOC2 puede hablar de cosas desconectadas

Guarded FOC2: cada existencial tiene que aparecer en un Edge positivo

$$Red(x) \wedge \exists y\big(Edge(x,y) \wedge \exists^{\geq 2} x[Edge(y,x) \wedge Blue(x)]\big)$$

Cada existencial aparece en una edge positivo (una guardia)



Este no tiene guardia

$$Red(x) \wedge \exists y\big(\neg Edge(x,y) \wedge \exists^{\geq 2} x[Edge(y,x) \wedge Blue(x)]\big)$$

Teorema (Barceló et al, 2020):

A función lógica de (grafo, nodo) a {0,1} puede ser computada por una MPNN solo si se puede expresar en guarded FOC2.

Demostración necesita algunas cosas de lógica,

Solo damos intuición en clases (Está también en el paper)

Teorema (Barceló et al, 2020):

A función lógica de (grafo, nodo) a {0,1} puede ser computada por una MPNN si y solo si se puede expresar en guarded FOC2.

Osea que las MPNN computan cualquier clasificador lógico en FOC2.
(Sin siquiera aproximar, lo computan directo).

Ejemplo de la construcción en pizarra (demostración en el paper).

Teorema (Barceló et al, 2020):

Toda función lógica de (grafo, nodo) a {0,1} que se puede expresar en FOC2 puede ser computada por una readout-MPNN.

readout-MPNN: Agregar una función de readout que agregue embeddings de todos los nodos, no solo los vecinos

Pero qué computan las MPNNs?

En funciones binarias se parecen a **guarded FOC2**.

Pero **MPNNs** pueden computar funciones arbitrarias!

Aproximación de funciones

Consideremos un set F de funciones.

La clausura de F contiene todas las funciones h tal que:

- **Hay una secuencia de funciones F**
- **Cada función nueva de la secuencia está mas cerca de h**

Idea: usando F podemos aproximar cualquier cosa en su clausura

Pero qué computan las MPNNs?

Theorem (Geerts y R., 2022):

Arquitecturas de MPNN pueden aproximar cualquier función tal que su separation power esté sobre el de Colour Refinement

(separation power)

$$(G, v, w) \in \rho(f) \Leftrightarrow f(G, v) = f(G, w)$$

Pero qué computan las MPNNs?

Toma una función f.

Supón que su separation power f está sobre el de colour refinement.

$$(G, v, w) \in \rho(f) \Leftrightarrow f(G, v) = f(G, w)$$

$$\rho(CR) \subseteq \rho(f)$$

But what can MPNNs compute?

Toma una función f.

Supón que su separation power f está sobre el de colour refinement.

$$(G, v, w) \in \rho(f) \Leftrightarrow f(G, v) = f(G, w)$$

$$\rho(CR) \subseteq \rho(f)$$

Luego f puede ser aproximado por alguna MPNN

(Siempre que se satisfagan algunas condiciones pequeñas)

Pero qué computan las MPNNs?

Theorem (Geerts y R., 2022):

Arquitecturas de MPNN pueden aproximar cualquier función tal que su separation power esté sobre el de Colour Refinement

Idea:

- Sea F todas las funciones que son computadas por una MPNN
- funciones en la clausura de F no pueden separar mejor que las de F
- Vimos que Colour refinement está en F

Asi… MPNNs

El modelo es barato

Asi… MPNNs

El modelo es barato

Las capas de MPNNs nos dan una buena forma, y enterable, de capturar la información de un grafo (mucho mejor que depender en la matriz de adyacencia!)

So… MPNNs

El modelo es barato

Las capas de MPNNs nos dan una buena forma, y enterable, de capturar la información de un grafo (mucho mejor que depender en la matriz de adyacencia!)

Pero, les falta expresividad:

- No cuentan triángulos
- No capturan información entre nodos bien lejos
- Las auto supervisadas tienen problemas peores (e.g. Rampášek et al.)

Indice

1. Poder de separación - repaso clase pasada
2. Qué hacen realmente las GNNs? Versión lógica
3. Qué hacen realmente las GNNs? Version funciones aproximadas

*** si queda tiempo (slides en inglés perdón)

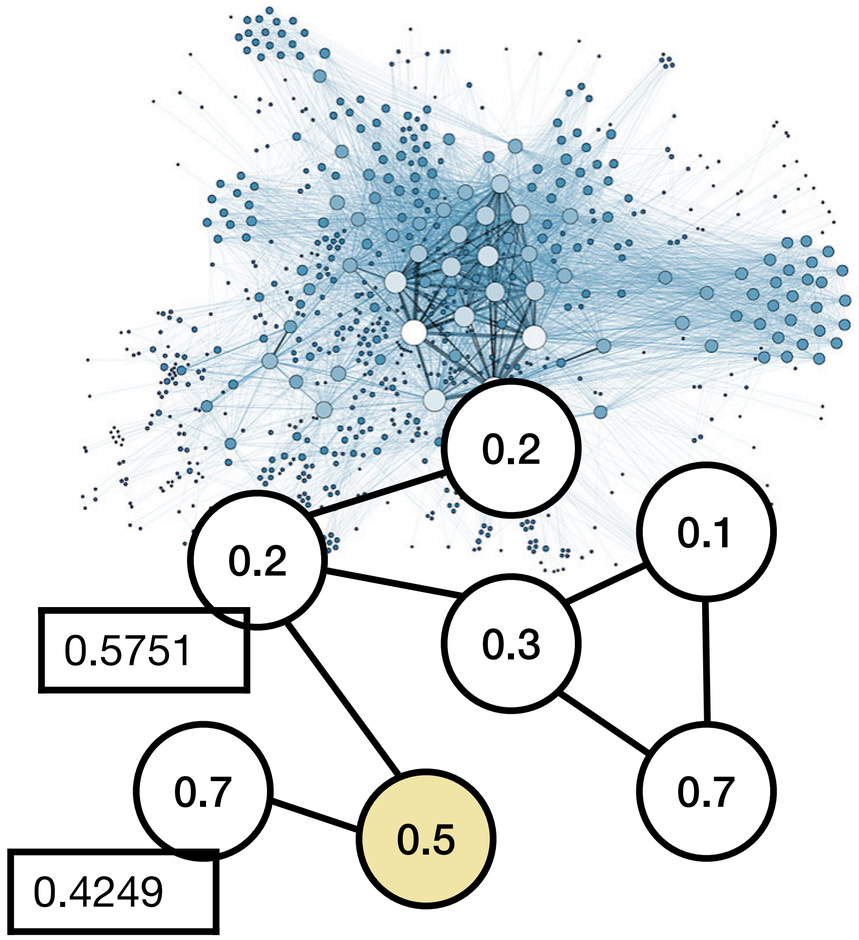
4. Beyond simple GNNs
5. How to study them

Beyond MPNNs

(Not comprehensive) survey on GNNs more expressive than MPNNs.

Goal is to understand architectural designs

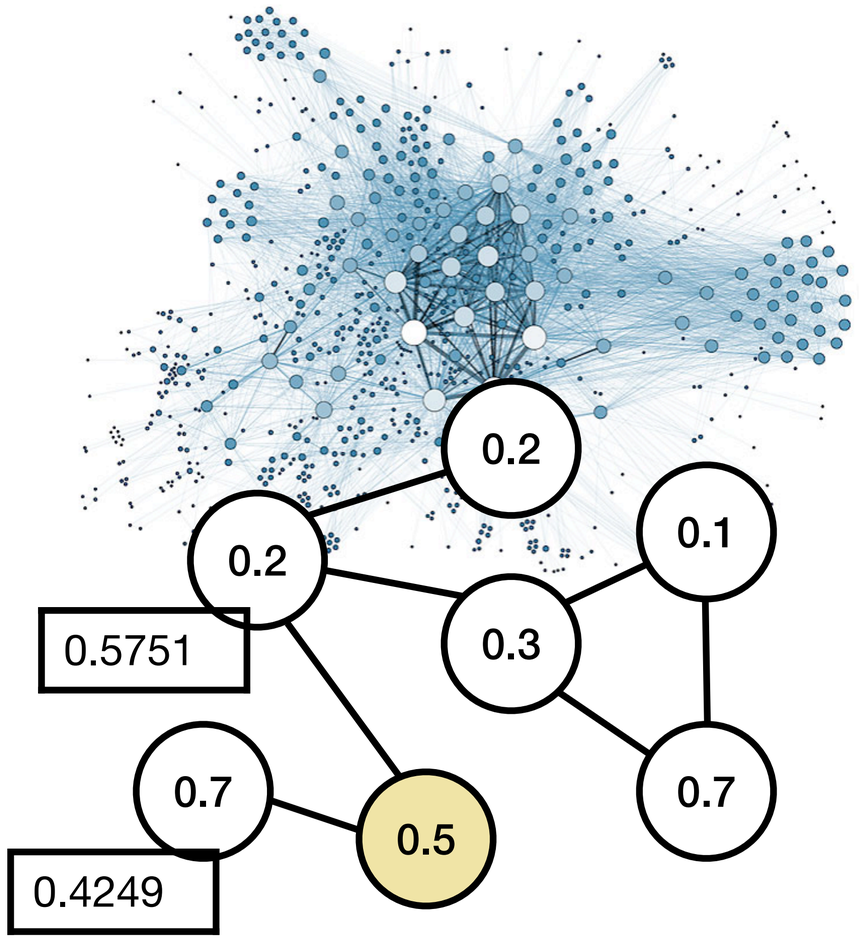
Beyond MPNNs



1. Attention (Veličković et al. 2017)

Attention weight applied to each neighbour before aggregation

Beyond MPNNs

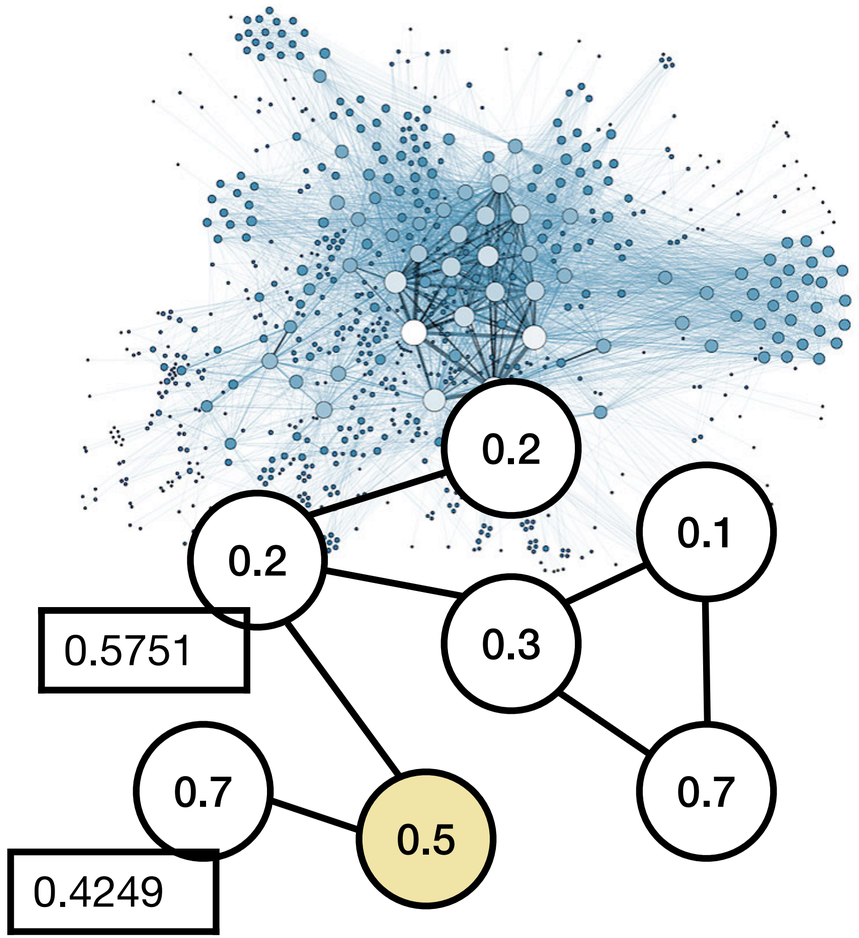


1. Attention (Veličković et al. 2017)

Attention weight applied to each neighbour before aggregation

Separation power bounded by Colour Refinement

Beyond MPNNs



2. Extra node/graph info

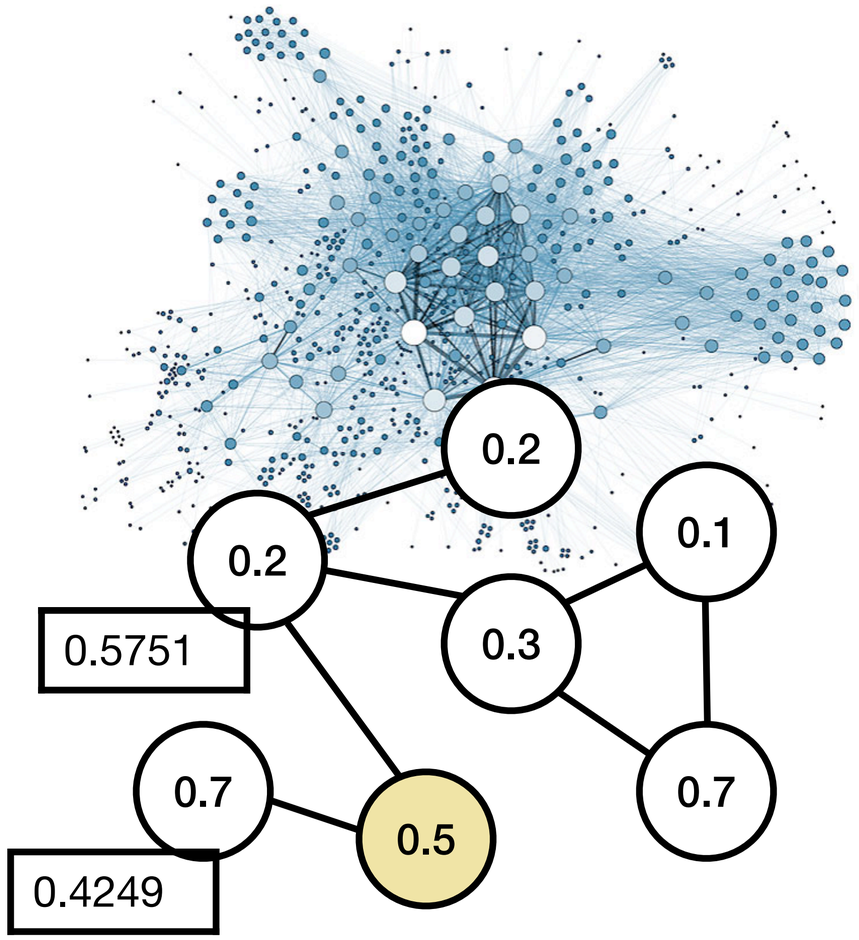
Subgraph info (e.g. Bouritsas et al. 2020, Barceló et al. 2021)

Spectral information (e.g. Kreuzer et al. 2021)

Pairwise info (e.g. Li et al. 2020)

…

Beyond MPNNs



2. Extra node/graph info

Subgraph info (e.g. Bouritsas et al. 2020, Barceló et al. 2021)

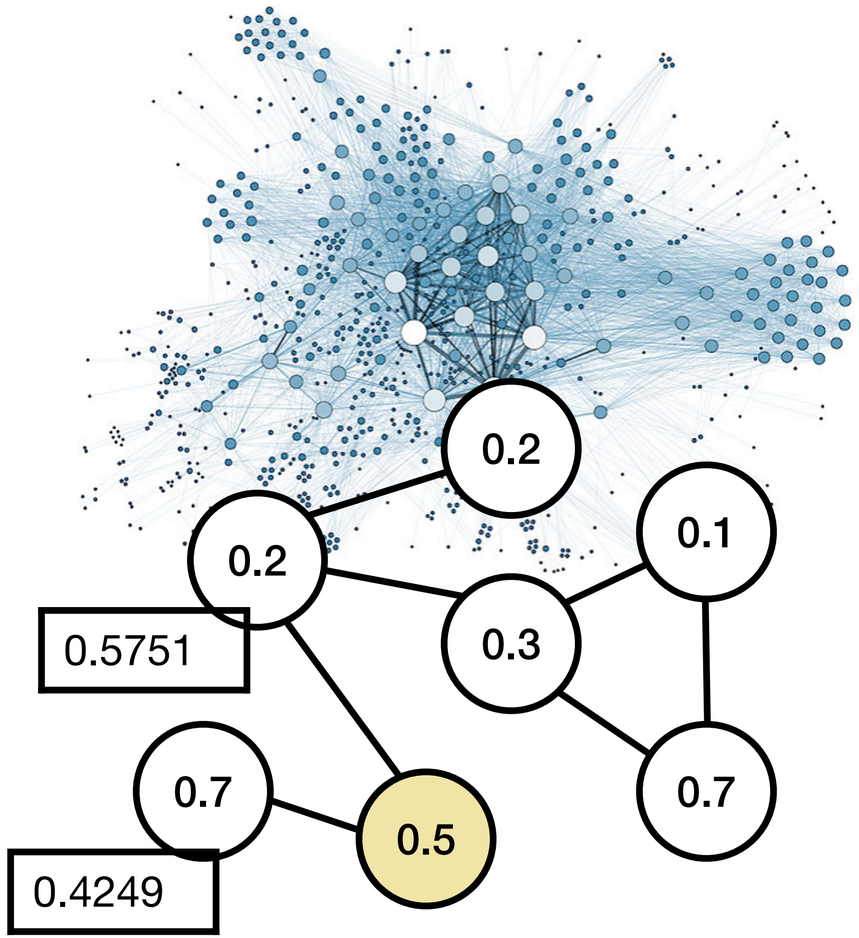
Spectral information (e.g. Kreuzer et al. 2021)

Pairwise info (e.g. Li et al. 2020)

…

They do increase power, but very dependent on design

Beyond MPNNs



3. Global Information

Add a virtual node that is connected to the entire graph (e.g. Cai et al. 2023)

They do seem to increase power as well, but many design questions left open.

Beyond MPNNs

Are there any general architectures that can put us beyond MPNNs?

Graph Transformers

Attention + **Extra graph info** + **More than neighbours**

Graph Transformers

Attention + **Extra graph info** + **More than neighbours**

Weights are combined with messages, they are trainable
and depend on the same / other parts of the graph.

Graph Transformers

Attention + **Extra graph info** + **More than neighbours**

Takes the role of positional encoding in text transformers.

Anything that we can pass to nodes so that they better understand how is the graph around them.

Graph Transformers

Attention + **Extra graph info** + **More than neighbours**

Either via attention or via more complex masks telling us
what node / edge information is aggregated at each step.

Graph Transformers

Attention + **Extra graph info** + **More than neighbours**

Several results for graph transformers show that they are **universal approximators**. But several practical issues remain open!

Example: GPS (Rampášek et al. 2022)

Attention + **Extra graph info** + **More than neighbours**

Using only node features and previous embeddings

Example: GPS (Rampášek et al. 2022)

Attention + **Extra graph info** + **More than neighbours**

Add all eigenvectors of the graph laplacian as features

One-hot encoding of edges

Example: GPS (Rampášek et al. 2022)

Attention + **Extra graph info** + **More than neighbours**

Structure of the MPNN is maintained, except they also use edge information in nodes

Example: GPS (Rampášek et al. 2022)

Attention + **Extra graph info** + **More than neighbours**

Structure of the MPNN is maintained, except they also use edge information in nodes

These transformers can approximate any function!

Example: GPS (Rampášek et al. 2022)

Attention + **Extra graph info** + **More than neighbours**

Structure of the MPNN is maintained, except they also use edge information in nodes

These transformers can approximate any function!

Proof follows from results on sparse text - to - text transformers (Yun et al. 2020, Yun et al. 2020, Zaheer et al. 2020)

Sparse text-to-text transformers, graphs

Graphs can be understood as sequence of text:
just give the adjacency list as a string

Then, mask selecting all neighbours -> mask of positions corresponding to those edges

Sparse text-to-text transformers, graphs

Graphs can be understood as sequence of text:
just give the adjacency list as a string

Then, mask selecting all neighbours -> mask of positions corresponding to those edges

From Yun et al. 2020, Zaheer et al. 2020:

If positions for each text token follow either
- **a star shape (one token attends to every other token)**
- **A hamiltonian graph (so one can always reach one token from another following this graph)**

Then text-to-text transformers can approximate any function.

Example: EXPHORMER (Shirzad et al 2023)

Attention + **Extra graph info** + **More than neighbours**

Messages are aggregated according to a mask computed from a random subgraph (with good properties) and some common global nodes

Example: EXPHORMER (Shirzad et al 2023)

Attention + **Extra graph info** + **More than neighbours**

Messages are aggregated according to a mask computed from a random subgraph (with good properties) and some common global nodes

These transformers can approximate any function!

Higher Order MPNNs

Higher Order MPNNs

As with MPNNs,

but we compute embeddings for pairs of nodes

(think of a graph transformer without attention, but whose mask is the entire graph)

Higher Order MPNNs

As with MPNNs,

but we compute embeddings for pairs of nodes

(think of a graph transformer without attention, but whose mask is the entire graph)

Power bounded by the
2-dimensional Weisfeiler-Lehman test

Higher Order MPNNs

As with MPNNs,

but we compute embeddings for k-tuples of nodes

(think of a graph transformer without attention, but whose mask is the entire graph)

Power bounded by the
k-dimensional Weisfeiler-Lehman test

Higher Order MPNNs

As with MPNNs,

but we compute embeddings for k-tuples of nodes

(think of a graph transformer without attention, but whose mask is the entire graph)

But extremely expensive to compute

Understanding Higher order MPNNs via their algebraic definition

Another tool to show expressive power of MPNN-based architectures.

From Geerts et al. 2022

Understanding Higher order MPNNs via their algebraic definition

- Here features assigned to tuples of vertices

$$\mathbf{v} = (v_1, \ldots, v_\ell)$$

$$(G, \mathbf{v}) \mapsto \varphi(G, \mathbf{v}) \in \mathbb{R}^d$$

- MPNNs use 1-vertex embeddings
- Most Higher order MPNNs rely on manipulating these embeddings

Layer by layer definition of vertex embeddings → **Using a procedural language**

→ **Instead of showing results for every MPNN architecture, show them for fragments of this language**

Layer by layer definition of vertex embeddings → **Using a procedural language**

Atomic operations	**One hot encodings, adjacency, equality/disequality**
Complex operations	**Function application (MLPs), aggregation**

Atomic operations

Label Retrieve a particular feature of a vector of node features

$$\mathsf{Lab}^{j}_{x_i} : (G, \mathbf{v}) \mapsto \mathsf{Lab}_j(G, v_i) \in \mathbb{R}$$

$$\mathbf{v} = (v_1, \ldots, v_\ell)$$

Atomic operations

Label Retrieve a particular feature of a vector of node features

$$\mathsf{Lab}^{j}_{x_i} : (G, \mathbf{v}) \mapsto \mathsf{Lab}_j(G, v_i) \in \mathbb{R}$$
$$\mathbf{v} = (v_1, \ldots, v_\ell)$$

Edge Boolean test for edge between two vertices

$$\mathsf{E}_{x_i,x_j} : (G, \mathsf{v}) \mapsto \begin{cases} 1 & \text{if } (v_i, v_j) \in E_G \\ 0 & \text{otherwise} \end{cases}$$

Atomic operations

Label — Retrieve a particular feature of a vector of node features

$$\mathsf{Lab}^{j}_{x_i} : (G, \mathbf{v}) \mapsto \mathsf{Lab}_j(G, v_i) \in \mathbb{R}$$
$$\mathbf{v} = (v_1, \ldots, v_\ell)$$

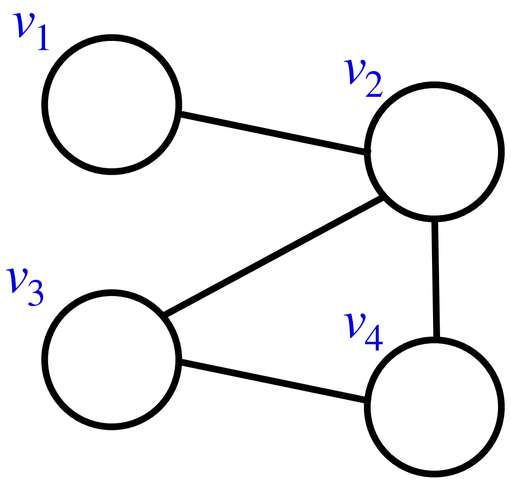
Edge — Boolean test for edge between two vertices

$$\mathsf{E}_{x_i,x_j} : (G, \mathsf{v}) \mapsto \begin{cases} 1 & \text{if } (v_i, v_j) \in E_G \\ 0 & \text{otherwise} \end{cases}$$

Dis(equality) — Boolean test if vertices are distinct/equal

$$\mathbf{1}_{x_i=x_j} : (G, \mathbf{v}) \mapsto \begin{cases} 1 & \text{if } v_i = v_j \in E_G \\ 0 & \text{otherwise} \end{cases} \qquad \mathbf{1}_{x_i\neq x_j} : (G, \mathbf{v}) \mapsto \begin{cases} 1 & \text{if } v_i \neq v_j \in E_G \\ 0 & \text{otherwise} \end{cases}$$

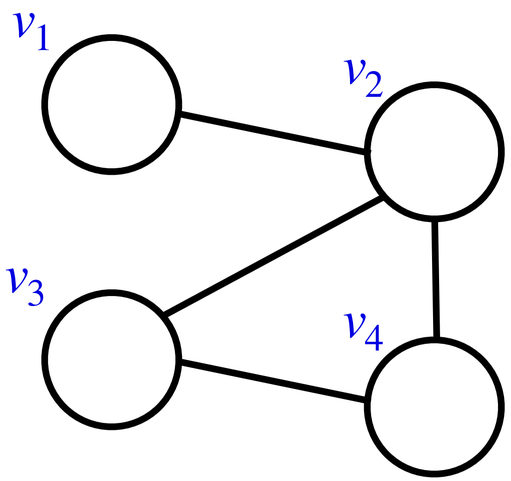
Aggregation



$$emb_{x_1 x_2}$$

$$(v_1, v_1) \mapsto 0.1$$
$$(v_1, v_2) \mapsto 0.5$$
$$(v_1, v_3) \mapsto 0.1$$
$$(v_1, v_4) \mapsto 0.5$$
$$\vdots$$

Aggregation



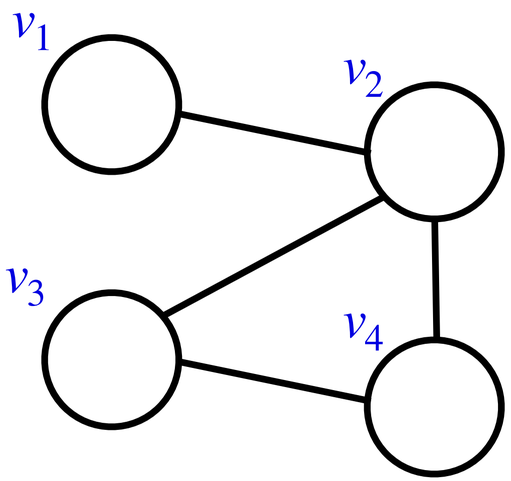
$emb_{x_1 x_2}$

$(v_1, v_1) \mapsto 0.1$
$(v_1, v_2) \mapsto 0.5$
$(v_1, v_3) \mapsto 0.1$
$(v_1, v_4) \mapsto 0.5$
$\vdots$

$\text{mean}_{x_2}(emb_{x_1 x_2} \mid \mathbf{1}_{x_1 = x_1})$

$(v_1) \mapsto 0.3$
$(v_2) \mapsto \ldots$
$\vdots$

Aggregation



$$emb_{x_1 x_2}$$

$$(v_1, v_1) \mapsto 0.1$$
$$(v_1, v_2) \mapsto 0.5$$
$$(v_1, v_3) \mapsto 0.1$$
$$(v_1, v_4) \mapsto 0.5$$
$$\vdots$$

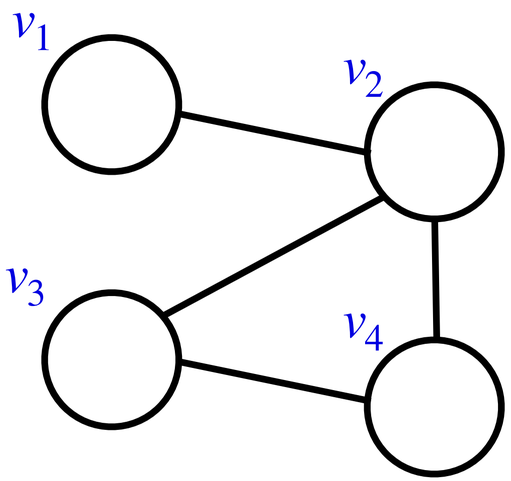
Aggregation is performed over second argument

$$\text{mean}_{x_2}(emb_{x_1 x_2} \mid \mathbf{1}_{x_1 = x_1})$$

$$(v_1) \mapsto 0.3$$
$$(v_2) \mapsto \ldots$$
$$\vdots$$

over all pairs satisfying this condition

Aggregation



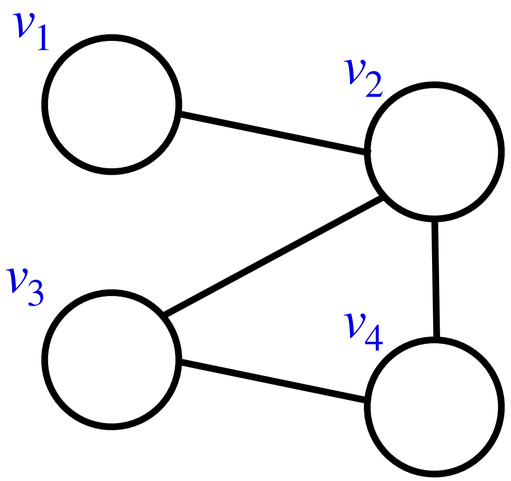
$emb_{x_1 x_2}$

$(v_1, v_1) \mapsto 0.1$
$(v_1, v_2) \mapsto 0.5$
$(v_1, v_3) \mapsto 0.1$
$(v_1, v_4) \mapsto 0.5$
$\vdots$

$\text{mean}_{x_2}(emb_{x_1 x_2} \mid E_{x_1, x_2})$

$(v_1) \mapsto 0.5$
$(v_2) \mapsto \ldots$
$\vdots$

Aggregation



$emb_{x_1 x_2}$

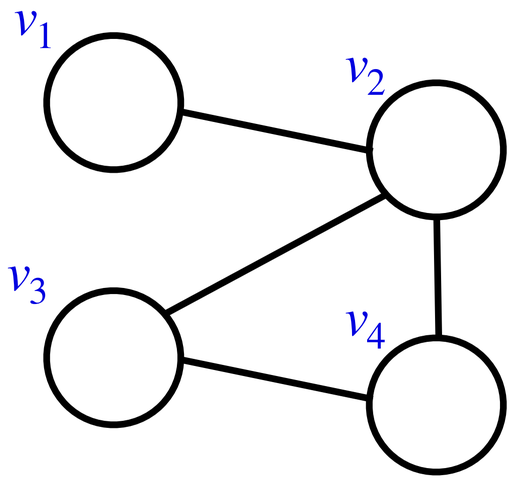
$(v_1, v_1) \mapsto 0.1$
$(v_1, v_2) \mapsto 0.5$
$(v_1, v_3) \mapsto 0.1$
$(v_1, v_4) \mapsto 0.5$
$\vdots$

$\text{mean}_{x_2}(emb_{x_1 x_2} \mid E_{x_1, x_2})$

$(v_1) \mapsto 0.5$
$(v_2) \mapsto \ldots$
$\vdots$

Only neighbours satisfy this

Function application



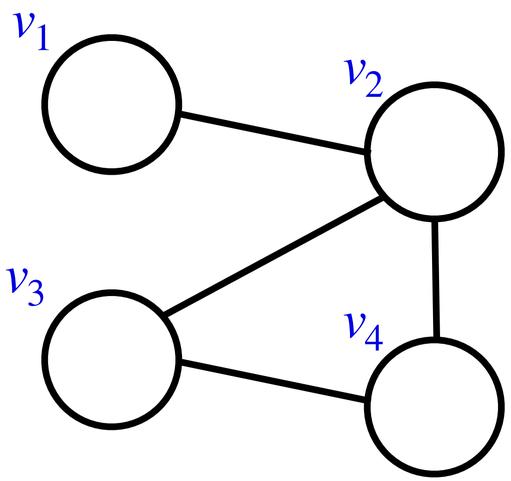
$$emb^1_{x_1x_2}$$

$$(v_1, v_1) \mapsto 0.1$$
$$(v_1, v_2) \mapsto 0.5$$
$$(v_1, v_3) \mapsto 0.1$$
$$(v_1, v_4) \mapsto 0.5$$
$$\vdots$$

$$emb^2_{x_2x_3}$$

$$(v_1, v_1) \mapsto 0.2$$
$$(v_1, v_2) \mapsto 0.2$$
$$(v_1, v_3) \mapsto 0.2$$
$$(v_1, v_4) \mapsto 0.2$$
$$\vdots$$

Function application



$$emb^1_{x_1x_2}$$

$$(v_1, v_1) \mapsto 0.1$$
$$(v_1, v_2) \mapsto 0.5$$
$$(v_1, v_3) \mapsto 0.1$$
$$(v_1, v_4) \mapsto 0.5$$
$$\vdots$$

$$emb^2_{x_2x_3}$$

$$(v_1, v_1) \mapsto 0.2$$
$$(v_1, v_2) \mapsto 0.2$$
$$(v_1, v_3) \mapsto 0.2$$
$$(v_1, v_4) \mapsto 0.2$$
$$\vdots$$

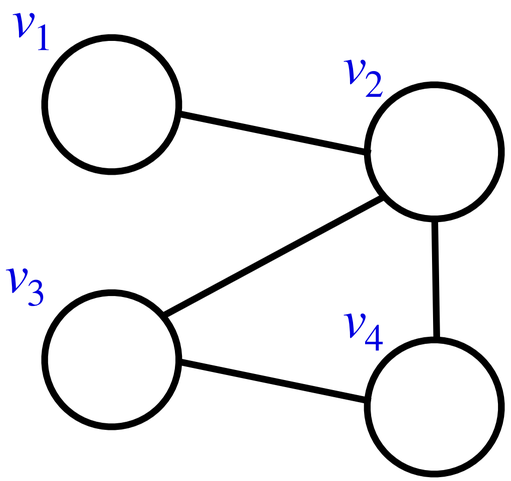
$$MLP(emb^1_{x_1x_2}, emb^2_{x_2,x_3})$$

$$\vdots$$

$$(v_1, v_2, v_3) = MLP(0.5,0.2) = 0.9$$

$$\vdots$$

Function application



$$emb^1_{x_1x_2}$$

$$(v_1, v_1) \mapsto 0.1$$
$$(v_1, v_2) \mapsto 0.5$$
$$(v_1, v_3) \mapsto 0.1$$
$$(v_1, v_4) \mapsto 0.5$$
$$\vdots$$

$$emb^2_{x_2x_3}$$

$$(v_1, v_1) \mapsto 0.2$$
$$(v_1, v_2) \mapsto 0.2$$
$$(v_1, v_3) \mapsto 0.2$$
$$(v_1, v_4) \mapsto 0.2$$
$$\vdots$$

Multi-layer perceptron

$$MLP(emb^1_{x_1x_2}, emb^2_{x_2,x_3})$$

$$\vdots$$

$$(v_1, v_2, v_3) = MLP(0.5,0.2) = 0.9$$

$$\vdots$$

Layer by layer definition of vertex embeddings

k-MPNNs: layered specification using only k+1 variables

Closure of all atomic operations under function application and aggregation

Layer by layer definition of vertex embeddings

k-MPNNs: layered specification using only k+1 variables

Theorem (Geerts et al. 2022):

The separation power of k-MPNNs is bounded by the k-dimensional WL algorithm

Layer by layer definition of vertex embeddings

k-MPNNs: layered specification using only k+1 variables

To understand the power of GNNS:

1- Write them as a K-MPNN
2- Use the theorem

Examples.

2-Folklore GNNs (Maron et. al 2020)

$$\varphi^{(t)}_{x_1,x_2} = \qquad \varphi^{(t-1)}_{x_1,x_3} \qquad \varphi^{(t-1)}_{x_3,x_2}$$

Examples.

2-Folklore GNNs (Maron et. al 2020)

$$\varphi^{(t)}_{x_1,x_2} = \qquad \mathrm{MLP}(\varphi^{(t-1)}_{x_1,x_3}) \cdot \mathrm{MLP}(\varphi^{(t-1)}_{x_3,x_2})$$

Examples.

2-Folklore GNNs (Maron et. al 2020)

$$\varphi^{(t)}_{x_1,x_2} = \mathrm{sum}_{x_3}(\mathrm{MLP}(\varphi^{(t-1)}_{x_1,x_3}) \cdot \mathrm{MLP}(\varphi^{(t-1)}_{x_3,x_2}) \mid \mathbf{1}_{x_1=x_1})$$

Examples.

2-Folklore GNNs (Maron et. al 2020)

$$\varphi^{(t)}_{x_1,x_2} = \mathrm{MLP}(\mathrm{sum}_{x_3}(\mathrm{MLP}(\varphi^{(t-1)}_{x_1,x_3}) \cdot \mathrm{MLP}(\varphi^{(t-1)}_{x_3,x_2}) \mid \mathbf{1}_{x_1=x_1}))$$

Function Approximation

Theorem (Geerts et al 2022):

k-MPNNs can approximate any function whose separation power is bounded by the k-dimensional WL algorithm

Function Approximation

Say you build a new GNN. What functions can it approximate?

1- Write them as a k-MPNN
2- Then it may only approximate functions bounded by k-WL

Function Approximation

Say you build a new GNN. What functions can it approximate?

1- Write them as a k-MPNN
2- Then it may only approximate functions bounded by k-WL

3- If the GNN has the same separation power as k-WL,
It approximates exactly those functions bounded by k-WL

Outline

1. how do GNNs work (+ some code)

2. Separation power of simple GNNs
3. What can they do

4. Beyond simple GNNs
5. How to study them

Concluding Remarks

Quest for the best way to include graph information to ML continues

But we are making huge progress!

Concluding Remarks

Quest for the best way to include graph information to ML continues

But we are making huge progress!

Big looming question:

Will there be any Large Graph Models?

What do we need for this? What are our shortcomings?