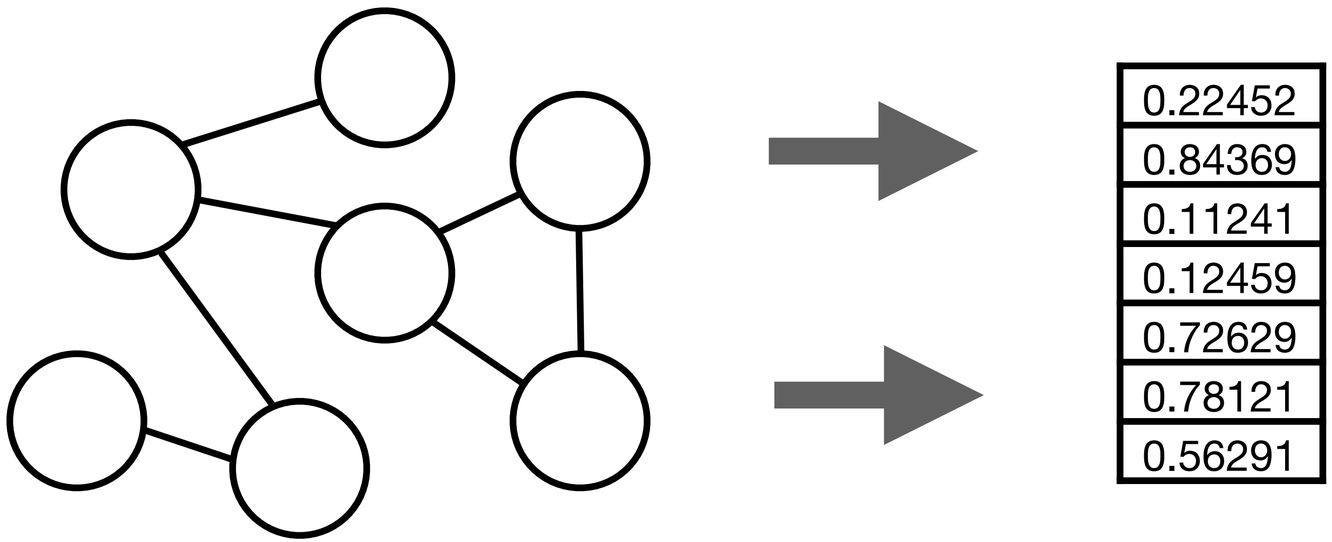
Understanding Graph Neural Networks

Juan Reutter D.

Universidad Católica de Chile
Instituto Milenio Fundamentos de los Datos

Graph Neural Networks

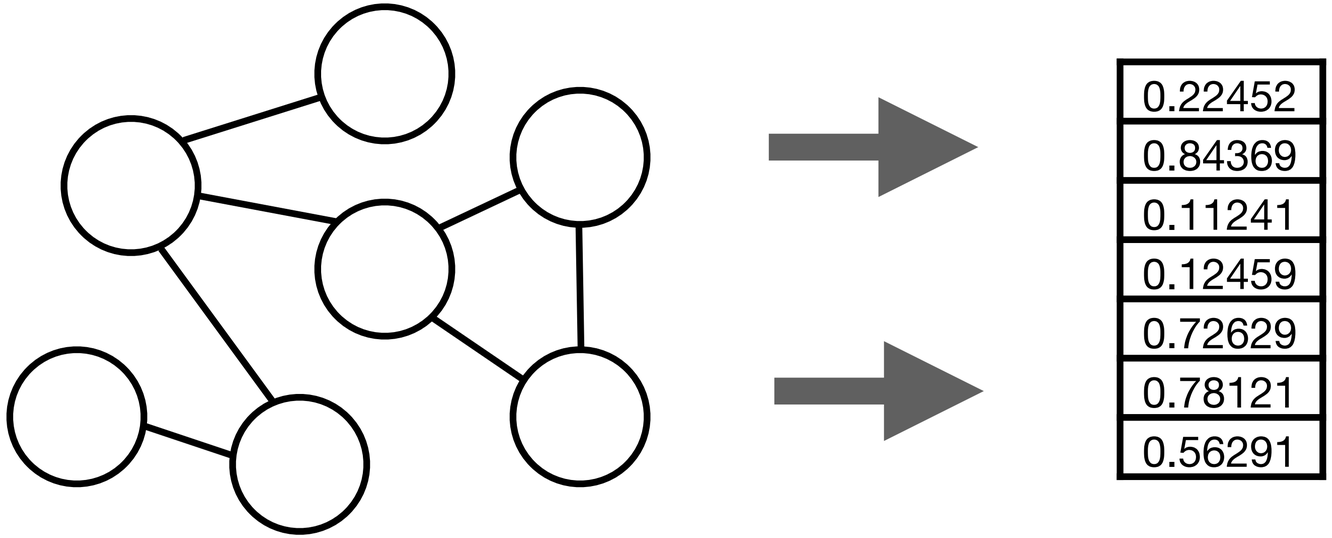
GNNs



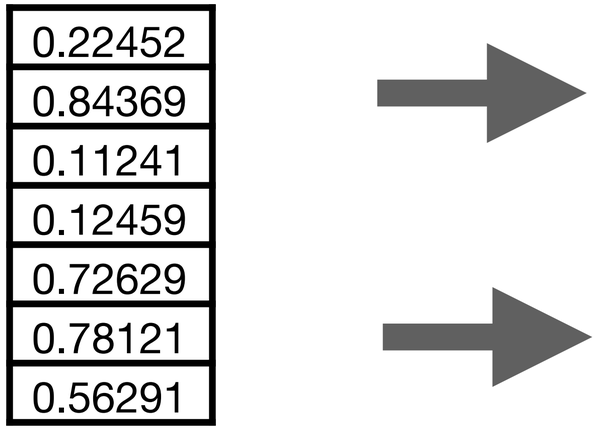
Learn graph embeddings

Graph Neural Networks

GNNs



Learn graph embeddings



Use them for ML tasks

Why graph embeddings?

Standard approach to graph learning problems

- Vertex Classification
- Link Prediction
- Regression of values in nodes
- (Sub)graph classification

Why graph embeddings?

Standard approach to graph learning problems

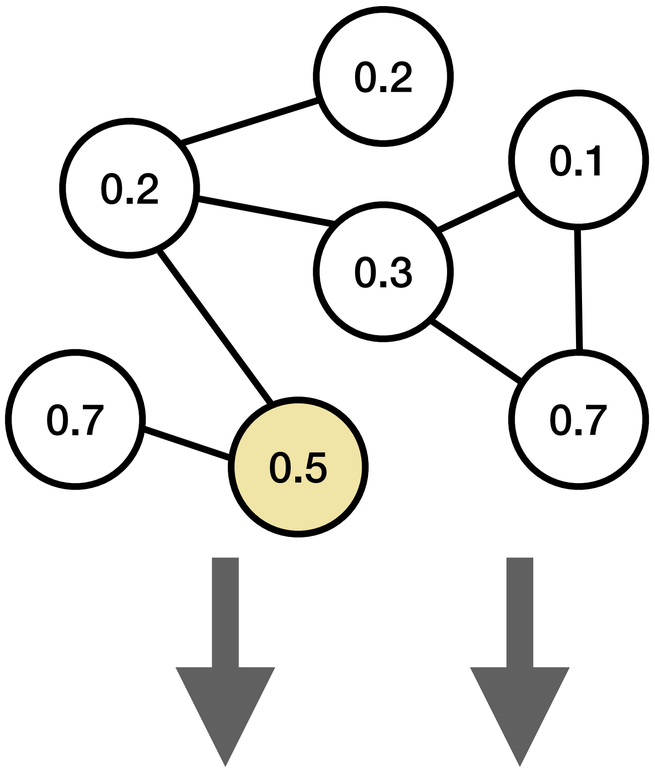
- Vertex Classification
- Link Prediction
- Regression of values in nodes
- (Sub)graph classification

Build embeddings, then apply known techniques

Embeddings combine knowledge + topology

- Nodes start with features
- Nodes in graphs are connected

- Embeddings “encode” topology



0.22	0.24	0.87
0.84	0.23	0.98
0.11	0.23	0.10
0.12	0.53	0.75
0.72	0.88	0.92
0.78	0.73	0.11
0.56	0.61	0.50

GNNs: add network data to ML tasks

- Bank database
- Need to predict credit turnout

users

id	full_name	enabled	last_login
1	John Smith	f	2017-10-25 10:26:10.015152
2	Alice Walker	t	2017-10-25 10:26:50.295461
3	Harry Potter	t	2017-10-25 10:26:50.295461
5	Jane Smith	t	2017-10-25 10:36:43.324015

checkouts

id	user_id	book_id	checkout_date	return_date
1	1	1	2017-10-15 14:43:18.095143-07	
2	1	2	2017-10-05 16:22:44.593188-07	2017-10-13 13:05:12.673382-05
3	2	2	2017-10-15 11:11:24.994973-07	2017-10-22 17:47:10.407569-07
4	5	3	2017-10-15 09:27:07.215217-07	

books

id	title	author	published_date	isbn
1	My First SQL book	Mary Parker	2012-02-22 12:08:17.320053-03	981483029127
2	My Second SQL book	John Mayer	1972-07-03 09:22:45.050088-07	857300923713
3	My Third SQL book	Cary Flint	2015-10-18 14:05:44.547516-07	523120967812

addresses

user_id	street	city	state
1	1 Market Stree	San Francisc	CA
2	2 Elm Street	San Francisc	CA
3	3 Main Street	Boston	MA

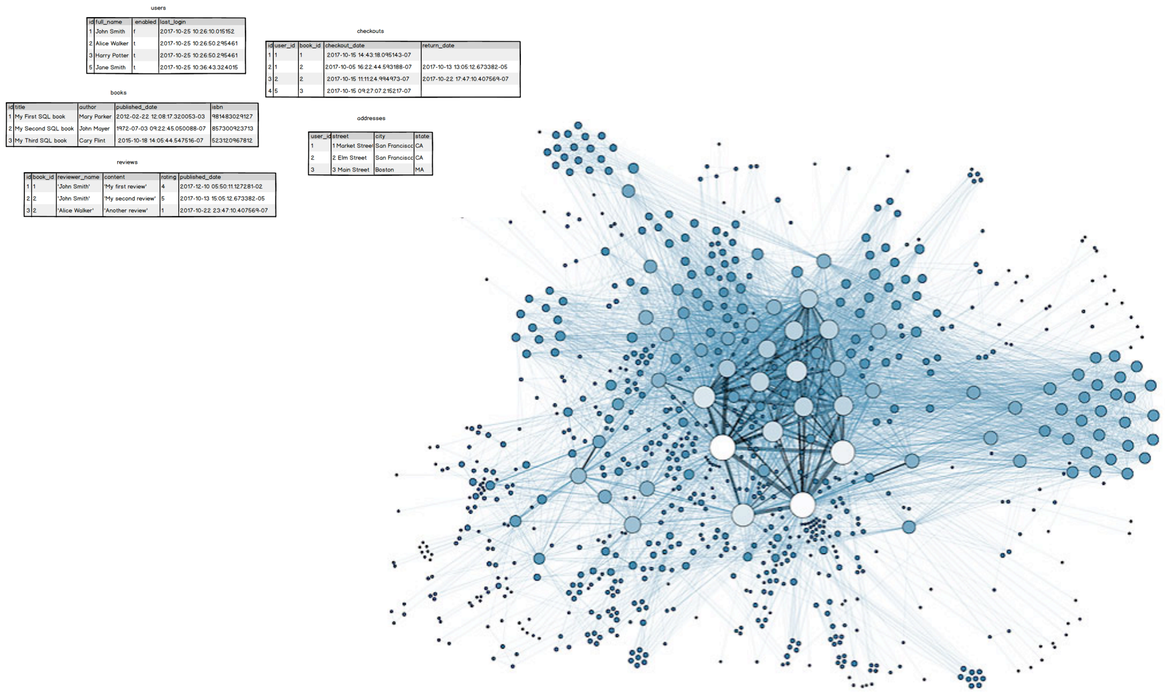
reviews

id	book_id	reviewer_name	content	rating	published_date
1	1	'John Smith'	'My first review'	4	2017-12-10 05:50:11.127281-02
2	2	'John Smith'	'My second review'	5	2017-10-13 15:05:12.673382-05
3	2	'Alice Walker'	'Another review'	1	2017-10-22 23:47:10.407569-07

Prediction

GNNs: add network data to ML tasks

- Bank database
- Need to predict credit turnout
- Incorporate social data?



Prediction

Image credit: AURAI

GNNs: add network data to ML tasks

- Bank database
- Need to predict credit turnout
- Incorporate social data?

users

id	full_name	enabled	last_login
1	John Smith	1	2017-10-25 10:26:10.015152
2	Alice Walker	1	2017-10-25 10:26:50.216461
3	Harry Potter	1	2017-10-25 10:26:50.216461
6	Jane Smith	1	2017-10-25 10:36:43.324015

checkouts

id	user_id	book_id	checkout_date	return_date
1	1	1	2017-10-15 14:43:18.016143-07	
2	1	2	2017-10-05 16:22:44.590188-07	2017-10-13 13:05:12.673382-05
3	2	2	2017-10-15 11:11:24.994173-07	2017-10-22 17:47:10.407569-07
4	5	3	2017-10-16 09:27:07.216217-07	

books

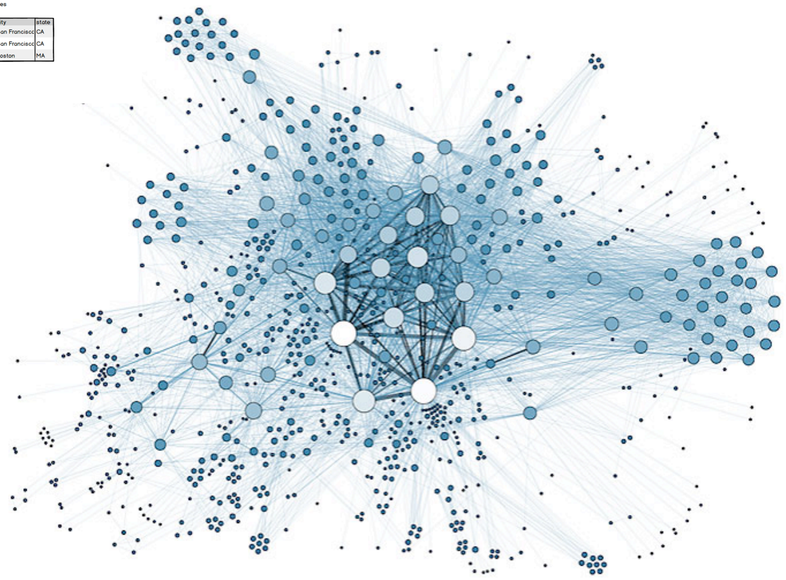
id	title	author	published_date	isbn
1	My First SQL book	Mary Parker	2012-02-22 12:08:17.320053-03	981483029127
2	My Second SQL book	John Mayer	1972-07-03 09:22:45.050088-07	857300923713
3	My Third SQL book	Cary Flint	2015-10-18 14:05:44.547516-07	623120967812

addresses

user_id	street	city	state
1	1 Market Street	San Francisco	CA
2	2 Elm Street	San Francisco	CA
3	3 Main Street	Boston	MA

reviews

id	book_id	reviewer_name	content	rating	published_date
1	1	'John Smith'	'My first review'	4	2017-12-10 05:50:11.127281-02
2	2	'John Smith'	'My second review'	5	2017-10-13 15:05:12.673382-05
3	2	'Alice Walker'	'Another review'	1	2017-10-22 23:47:10.407569-07



Prediction

How do we do this?

Image credit: AURAI

Can’t just add the data to the ML model

Idea: add table indicating edges in graph data.

run standard deep learning model

users

id	full_name	enabled	last_login
1	John Smith	f	2017-10-25 10:26:10.015152
2	Alice Walker	t	2017-10-25 10:26:50.216461
3	Harry Potter	t	2017-10-25 10:26:50.216461
5	Jane Smith	t	2017-10-25 10:36:43.324015

checkouts

id	user_id	book_id	checkout_date	return_date
1	1	1	2017-10-15 14:43:18.016143-07	
2	1	2	2017-10-05 16:22:44.590188-07	2017-10-13 13:05:12.673382-05
3	2	2	2017-10-16 11:51:24.994973-07	2017-10-22 17:47:10.407569-07
4	5	3	2017-10-16 09:27:07.215217-07	

books

id	title	author	published_date	isbn
1	My First SQL book	Mary Parker	2012-02-22 12:08:17.320053-03	981483029127
2	My Second SQL book	John Mayer	1972-07-03 09:22:45.050088-07	857300923713
3	My Third SQL book	Cary Flint	2015-10-18 14:05:44.547516-07	623120967812

addresses

user_id	street	city	state
1	1 Market Street	San Francisco	CA
2	2 Elm Street	San Francisco	CA
3	3 Main Street	Boston	MA

reviews

id	book_id	reviewer_name	content	rating	published_date
1	1	'John Smith'	'My first review'	4	2017-12-10 05:50:11.127281-02
2	2	'John Smith'	'My second review'	5	2017-10-13 15:05:12.673382-05
3	2	'Alice Walker'	'Another review'	1	2017-10-22 23:47:10.407569-07

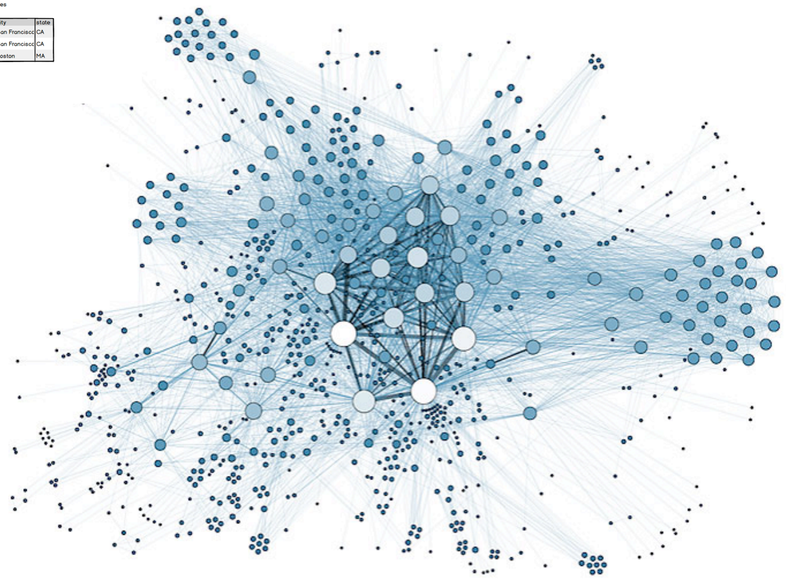


Image credit: AURAI

Can’t just add the data to the ML model

Idea: add table indicating edges in graph data.

run standard deep learning model

users

id	full_name	enabled	last_login
1	John Smith	f	2017-10-25 10:26:10.015152
2	Alice Walker	t	2017-10-25 10:26:50.216461
3	Harry Potter	t	2017-10-25 10:26:50.216461
5	Jane Smith	t	2017-10-25 10:36:43.324015

checkouts

id	user_id	book_id	checkout_date	return_date
1	1	1	2017-10-15 14:43:18.016143-07	
2	1	2	2017-10-05 16:22:44.590188-07	2017-10-13 13:05:12.673382-05
3	2	2	2017-10-16 11:11:24.994973-07	2017-10-22 17:47:10.407569-07
4	5	3	2017-10-16 09:27:07.215217-07	

books

id	title	author	published_date	isbn
1	My First SQL book	Mary Parker	2012-02-22 12:08:17.320053-03	981483029127
2	My Second SQL book	John Mayer	1972-07-03 09:22:45.050088-07	857300923713
3	My Third SQL book	Cary Flint	2015-10-18 14:05:44.547516-07	623120967812

addresses

user_id	street	city	state
1	1 Market Street	San Francisco	CA
2	2 Elm Street	San Francisco	CA
3	3 Main Street	Boston	MA

reviews

id	book_id	reviewer_name	content	rating	published_date
1	1	'John Smith'	'My first review'	4	2017-12-10 05:50:11.127281-02
2	2	'John Smith'	'My second review'	5	2017-10-13 15:05:12.673382-05
3	2	'Alice Walker'	'Another review'	1	2017-10-22 23:47:10.407569-07

This guy may have a lot of friends (unbounded number)

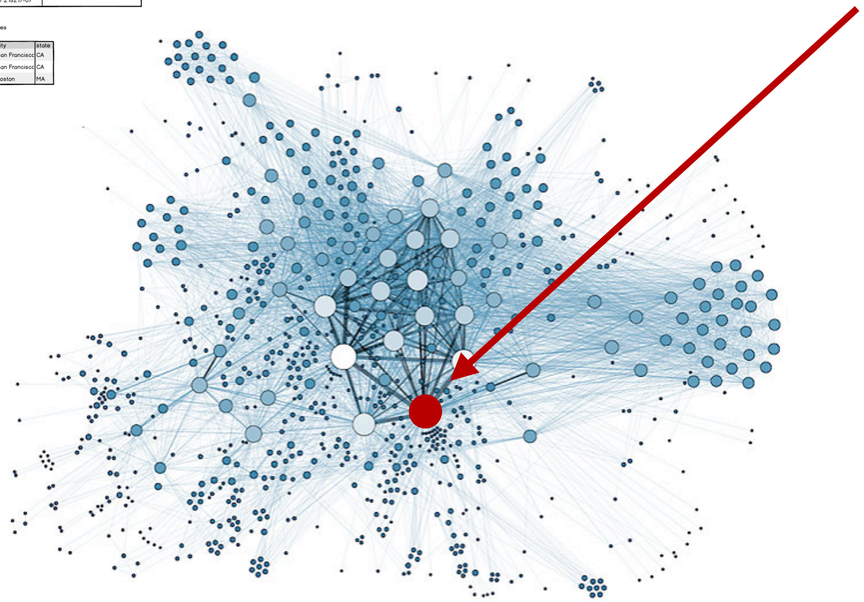


Image credit: AURAI

Can’t just add the data to the ML model

Idea: add table indicating edges in graph data.

run standard deep learning model

users

id	full_name	enabled	last_login
1	John Smith	f	2017-10-25 10:26:10.015152
2	Alice Walker	t	2017-10-25 10:26:50.216461
3	Harry Potter	t	2017-10-25 10:26:50.216461
5	Jane Smith	t	2017-10-25 10:36:43.324015

checkouts

id	user_id	book_id	checkout_date	return_date
1	1	1	2017-10-15 14:43:18.016143-07	
2	1	2	2017-10-05 16:22:44.590188-07	2017-10-13 13:05:12.673382-05
3	2	2	2017-10-16 11:11:24.994973-07	2017-10-22 17:47:10.407569-07
4	5	3	2017-10-16 09:27:07.215217-07	

books

id	title	author	published_date	isbn
1	My First SQL book	Mary Parker	2012-02-22 12:08:17.320053-03	981483029127
2	My Second SQL book	John Mayer	1972-07-03 09:22:45.050088-07	857300923713
3	My Third SQL book	Cary Flint	2015-10-18 14:05:44.547516-07	623120967812

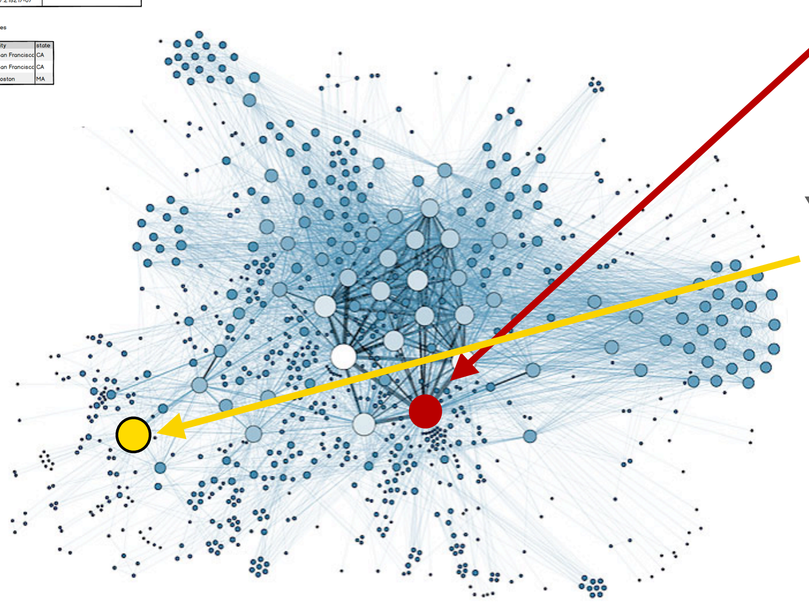
addresses

user_id	street	city	state
1	1 Market Street	San Francisco	CA
2	2 Elm Street	San Francisco	CA
3	3 Main Street	Boston	MA

reviews

id	book_id	reviewer_name	content	rating	published_date
1	1	'John Smith'	'My first review'	4	2017-12-10 05:50:11.127281-02
2	2	'John Smith'	'My second review'	5	2017-10-13 15:05:12.673382-05
3	2	'Alice Walker'	'Another review'	1	2017-10-22 23:47:10.407569-07

This guy may have a lot of friends (unbounded number)



Yellow node is important for prediction in red
But not directly connected

Image credit: AURAI

Can’t just add the data to the ML model

Idea: add table indicating edges in graph data.

run standard deep learning model

users

id	full_name	enabled	last_login
1	John Smith	f	2017-10-25 10:26:10.015152
2	Alice Walker	t	2017-10-25 10:26:50.216461
3	Harry Potter	t	2017-10-25 10:26:50.216461
5	Jane Smith	t	2017-10-25 10:36:43.324015

checkouts

id	user_id	book_id	checkout_date	return_date
1	1	1	2017-10-15 14:43:18.016143-07	
2	1	2	2017-10-05 16:22:44.590188-07	2017-10-13 13:05:12.673382-05
3	2	2	2017-10-16 11:11:24.994973-07	2017-10-22 17:47:10.407569-07
4	5	3	2017-10-16 09:27:07.215217-07	

books

id	title	author	published_date	isbn
1	My First SQL book	Mary Parker	2012-02-22 12:08:17.320053-03	981483029127
2	My Second SQL book	John Mayer	1972-07-03 09:22:45.050088-07	857300923713
3	My Third SQL book	Cary Flint	2015-10-18 14:05:44.547516-07	623120967812

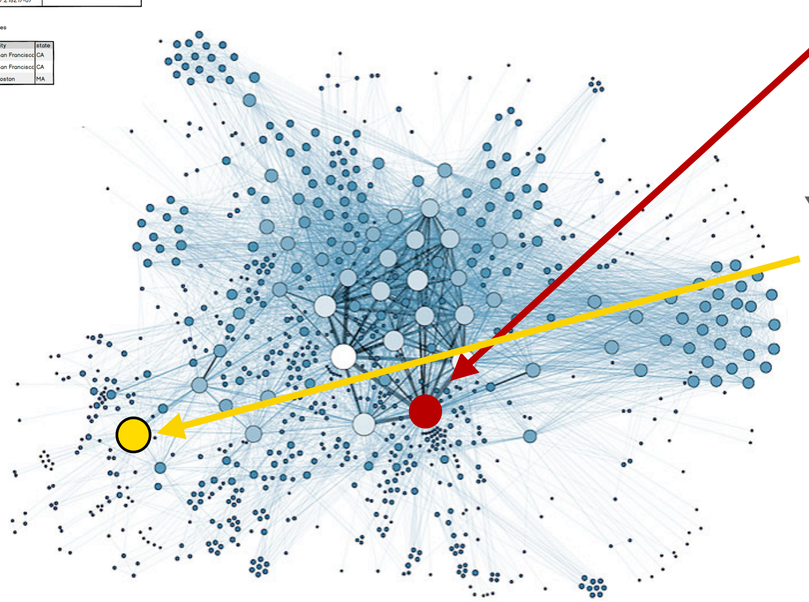
addresses

user_id	street	city	state
1	1 Market Street	San Francisco	CA
2	2 Elm Street	San Francisco	CA
3	3 Main Street	Boston	MA

reviews

id	book_id	reviewer_name	content	rating	published_date
1	1	'John Smith'	'My first review'	4	2017-12-10 05:50:11.127281-02
2	2	'John Smith'	'My second review'	5	2017-10-13 15:05:12.673382-05
3	2	'Alice Walker'	'Another review'	1	2017-10-22 23:47:10.407569-07

This guy may have a lot of friends (unbounded number)



Yellow node is important for prediction in red
But not directly connected

N individuals -> N^2 adjacency matrix!

Image credit: AURAI

Can’t just add the data to chatGPT either

GPT-4 has a 32000 token window

DBLP has 16,689,230 edges

Image credit: AURAI

Can’t just add the data to chatGPT either

GPT-4 has a 32000 token window

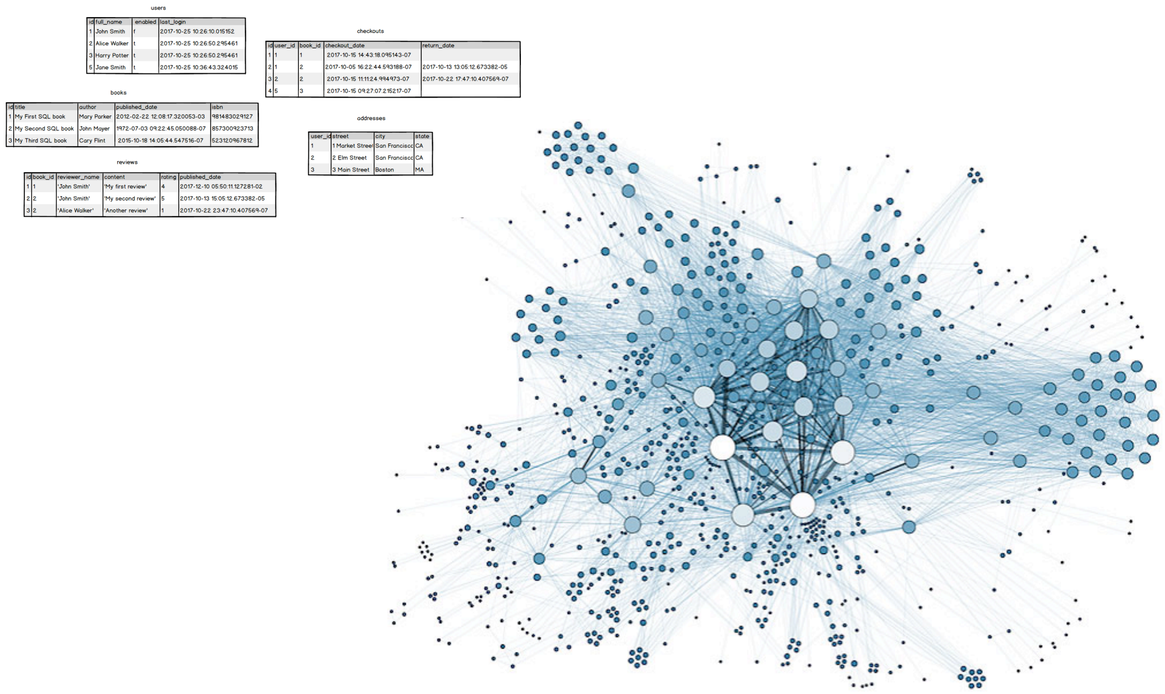
DBLP has 16,689,230 edges

Also: we really want our model to ignore the order in which nodes are passed

Image credit: AURAI

GNNs: add network data to ML tasks

- Bank database
- Need to predict credit turnout
- Incorporate social data?



Prediction

How do we do this? GNNs!

Image credit: AURAI

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

How basic GNN architectures look like

Guidelines for this tutorial

- Compute embeddings for nodes (maybe edges)
- Focus on one big graph (not several small graphs)
- Supervised / Self supervised embeddings (encoder/decoder training)
- Invariant to the way nodes/edges are passed or stored

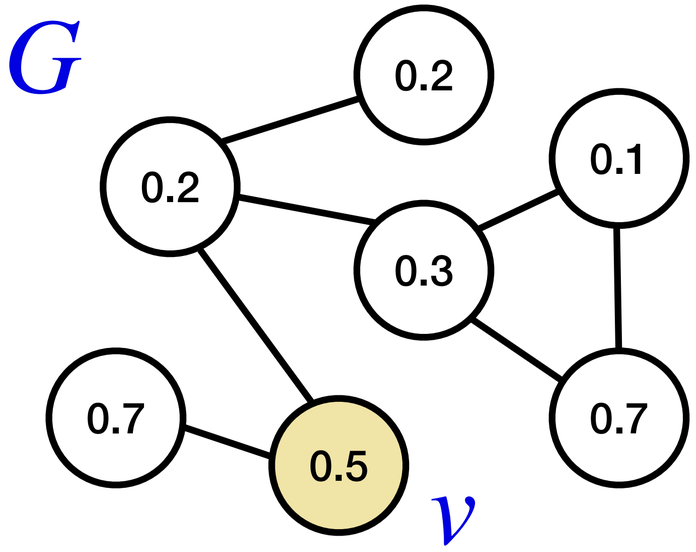
How basic GNN architectures look like

The foundation of GNNs is known as a Message Passing layer.

Architectures using only message passing are

Message Passing Neural Networks (MPNNs)

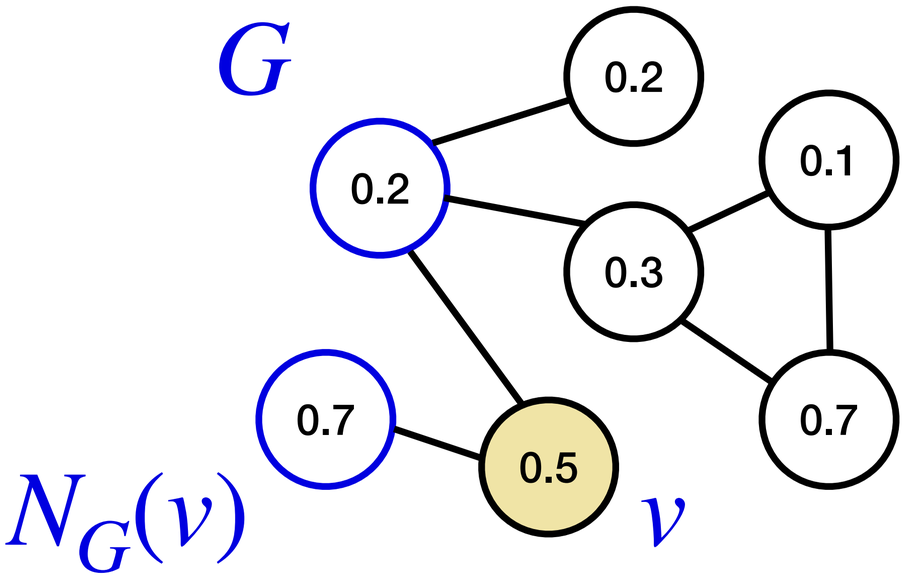
MPNN (Message Passing Neural Network) layer



0.22	0.24	0.87
0.84	0.23	0.98
0.11	0.23	0.10
0.12	0.53	0.75
0.72	0.88	0.92
0.78	0.73	0.11
0.56	0.61	0.50

$emb_G(v)$

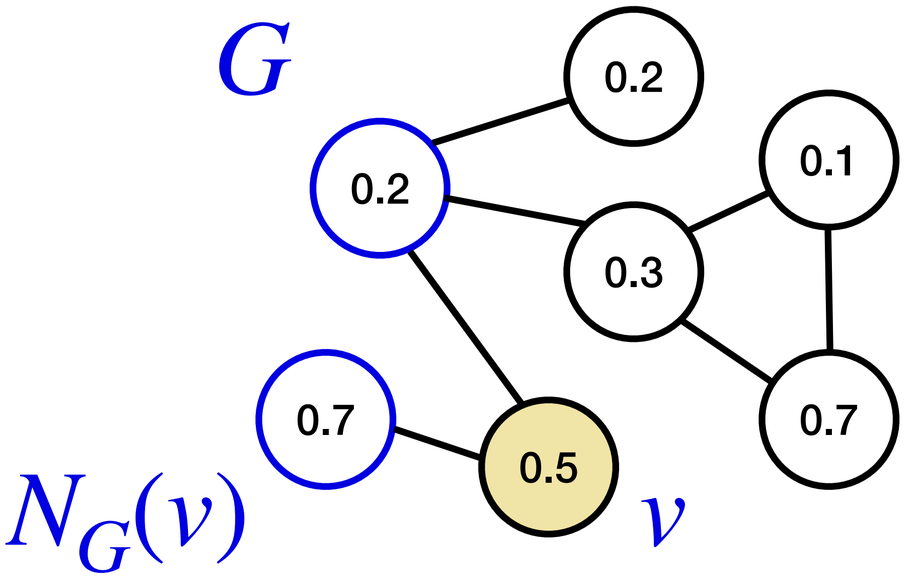
MPNN (Message Passing Neural Network) layer



0.22	0.24	0.87
0.84	0.23	0.98
0.11	0.23	0.10
0.12	0.53	0.75
0.72	0.88	0.92
0.78	0.73	0.11
0.56	0.61	0.50

$emb_G(v)$

MPNN (Message Passing Neural Network) layer



0.22	0.24	0.87
0.84	0.23	0.98
0.11	0.23	0.10
0.12	0.53	0.75
0.72	0.88	0.92
0.78	0.73	0.11
0.56	0.61	0.50

$emb_G(v)$

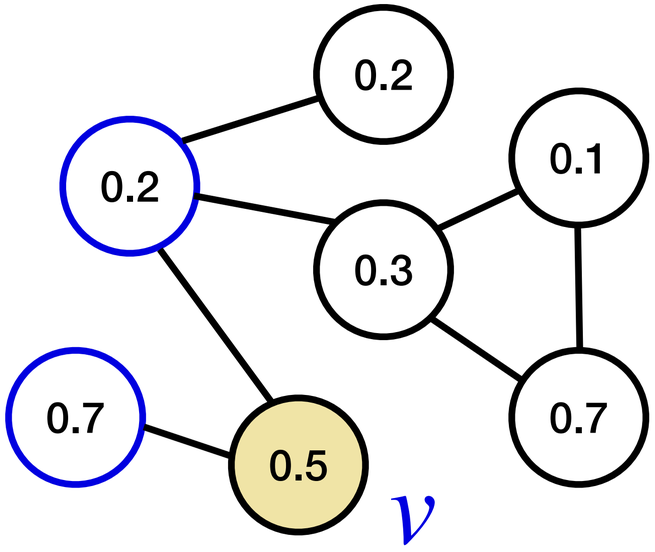
New embeddings $emb_G(v)$

from old embeddings $emb'_G(v)$

MPNN (Message Passing Neural Network) layer

New embeddings $emb_G(v)$
from old embeddings $emb'_G(v)$

$$\text{Aggregate}\left(\{\{\, emb'_G(w) \mid w \in N_G(v) \,\}\}\right)$$



$emb_G(v)$ will use $\text{Aggregate}(0.7, 0.2)$

MPNN (Message Passing Neural Network) layer

New embeddings $emb_G(v)$
from old embeddings $emb'_G(v)$

$$\text{Aggregate}\big(\{\{\, emb'_G(w) \mid w \in N_G(v) \,\}\}\big)$$

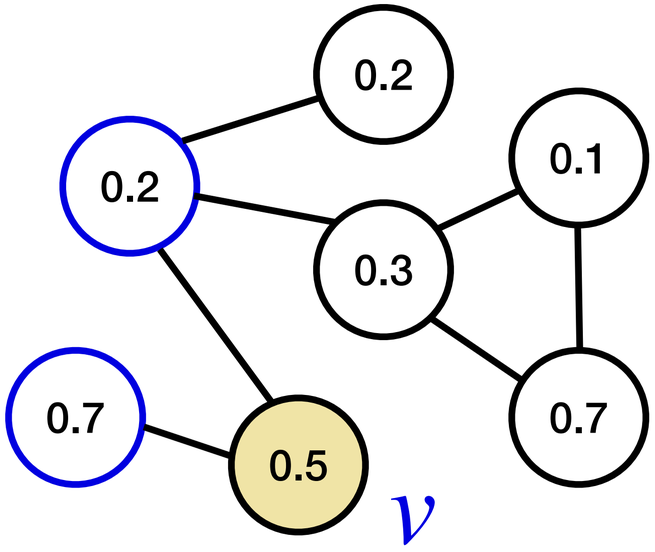
Aggregate function that aggregates embeddings of neighbours.
¡Must be permutation invariant!

Sum, Max, Avg, etc. Usually incorporates trained weights.

MPNN (Message Passing Neural Network) layer

New embeddings $emb_G(v)$
from old embeddings $emb'_G(v)$

$$\text{Aggregate}\big(\{\{\, emb'_G(w) \mid w \in N_G(v) \,\}\}\big)$$



$emb_G(v)$ will use $\text{Sum}\big(MLP(0.7), MLP(0.2)\big)$

MPNN (Message Passing Neural Network) layer

New embeddings $emb_G(v)$
from old embeddings $emb'_G(v)$

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

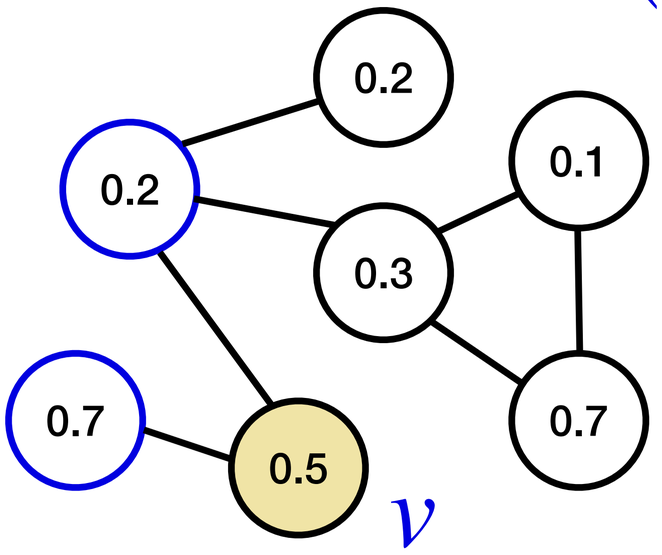
Update function that combines node embedding with aggregates.

Concat, Sum, etc. Usually incorporates trained weights.

MPNN (Message Passing Neural Network) layer

New embeddings $emb_G(v)$
from old embeddings $emb'_G(v)$

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



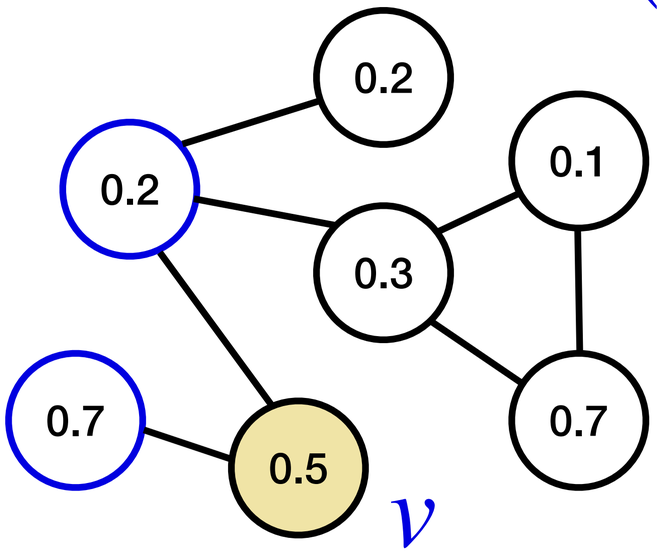
$emb_G(v)$ will use :

$$\text{Update}\Big(0.5, \text{Aggregate}(0.7, 0.2)\Big)$$

MPNN (Message Passing Neural Network) layer

New embeddings $emb_G(v)$
from old embeddings $emb'_G(v)$

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



$emb_G(v)$ will use :

$$\text{Update}\Big(MLP(0.5), \text{Sum}\big(MLP(0.7), MLP(0.2)\big)\Big)$$

MPNN (Message Passing Neural Network) layer

New embeddings $emb_G(v)$
from old embeddings $emb'_G(v)$

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

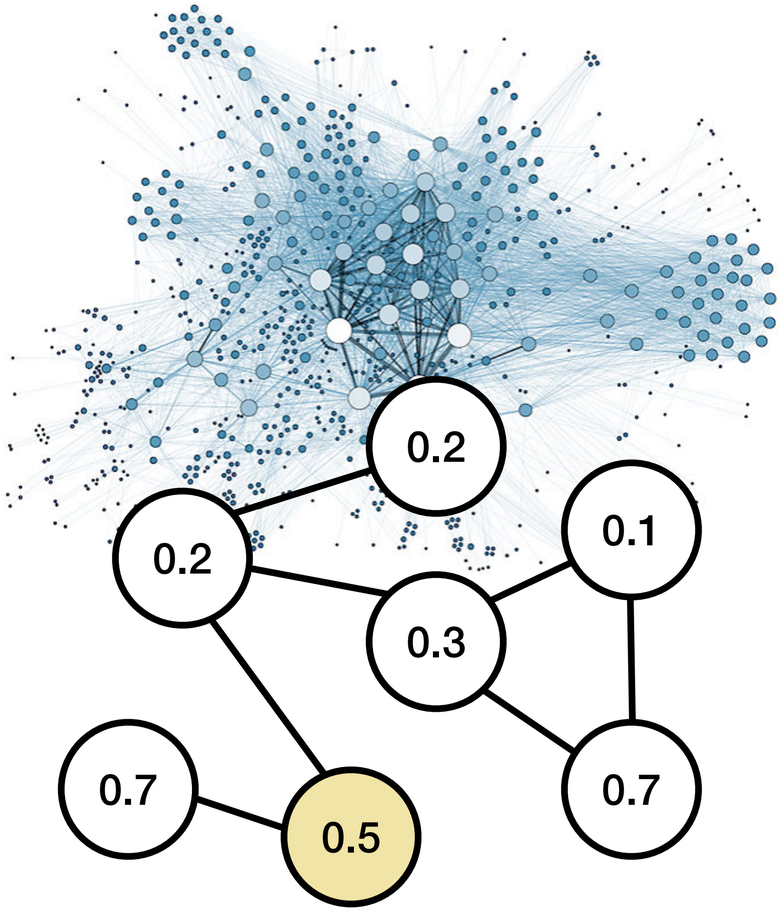
Non linear activation function for the result of aggregate - update

MPNN (Message Passing Neural Network) layer

New embeddings $emb_G(v)$
from old embeddings $emb'_G(v)$

1. Aggregate information from neighbours of nodes
2. Update this information with my old embedding
3. Pass it through a non linear activation function

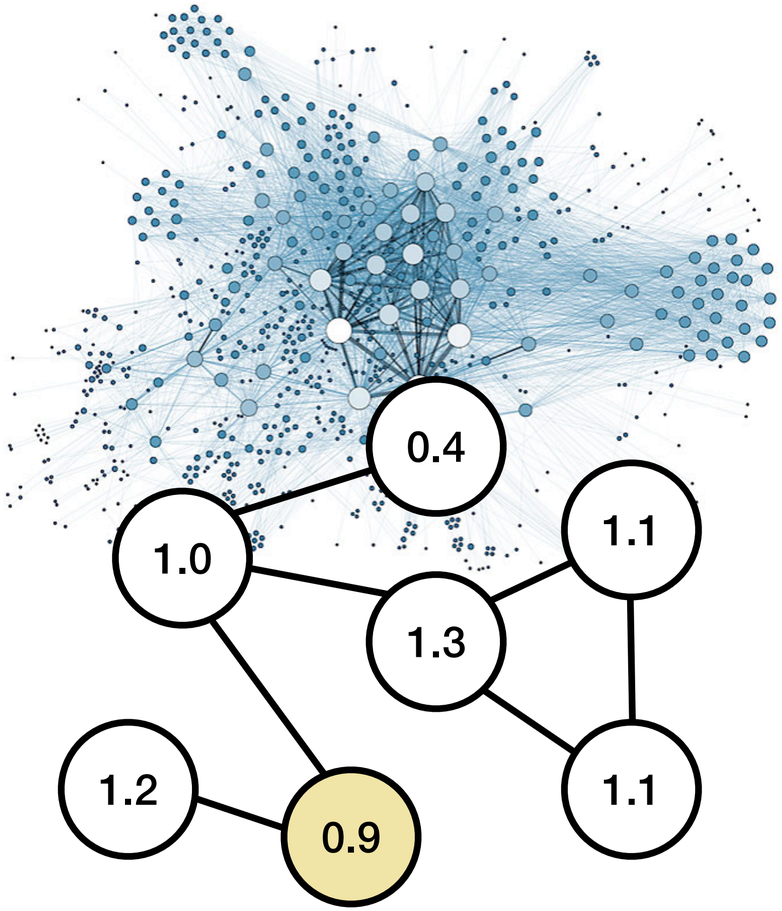
MPNN (Message Passing Neural Network) layer



Forward Propagation

- Nodes start with features
- Layer i: all nodes update according to embeddings layer (i-1)

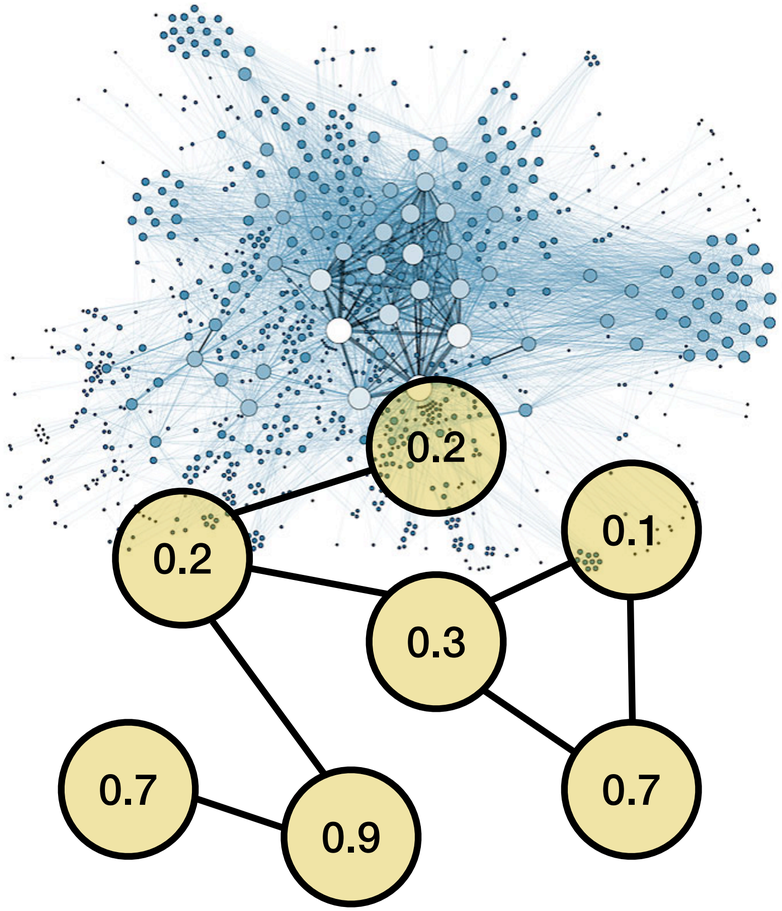
MPNN (Message Passing Neural Network) layer



Forward Propagation

- Nodes start with features
- Layer i: all nodes update according to embeddings layer (i-1)

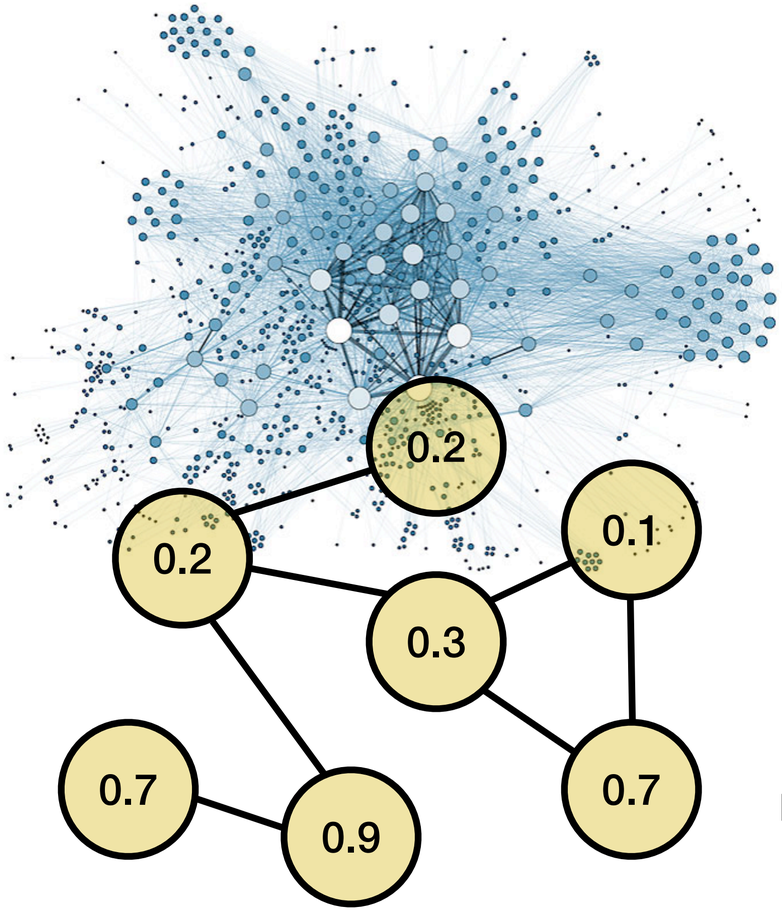
MPNN (Message Passing Neural Network) layer



Forward Propagation

- Nodes start with features
- Layer i: all nodes update according to embeddings layer (i-1)

Training MPNN layers (supervised case)



Forward Propagation

- Nodes start with features
- Layer i: all nodes update according to embeddings layer (i-1)
- Final layer: classification / regression task (at a node level)

Prediction

Training MPNN layers (supervised case)

Nodes
1
2
3
4
...
...
...
1000020
1000021
1000022

Training	Value
1	1
2	0
4	0
...	...
1000020	1
1000022	1

- For some nodes (training set) we know the correct value.
- Forward Propagation over all nodes.
- Only back propagate according to the training set.

Training MPNN layers (supervised case)

Nodes
1
2
3
4
...
...
...
1000020
1000021
1000022

Training	Value
1	1
2	0
4	0
...	...
1000020	1
1000022	1

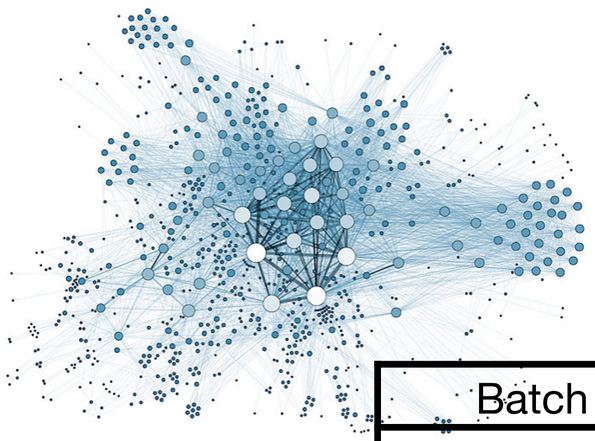
- For some nodes (training set) we know the correct value.
- Forward Propagation over all nodes.
- Only back propagate according to the training set.

Training	Value
1	0.84369
2	0.11241
4	0.12459
...	...
1000020	0.78121
1000022	0.56291

v/s

Y
1
0
0
...
1
1

Training MPNN layers (supervised case) + batching (Hamilton et al. 2017)



Training	Value
1	1
2	0
4	0
...	...
1000020	1
1000022	1

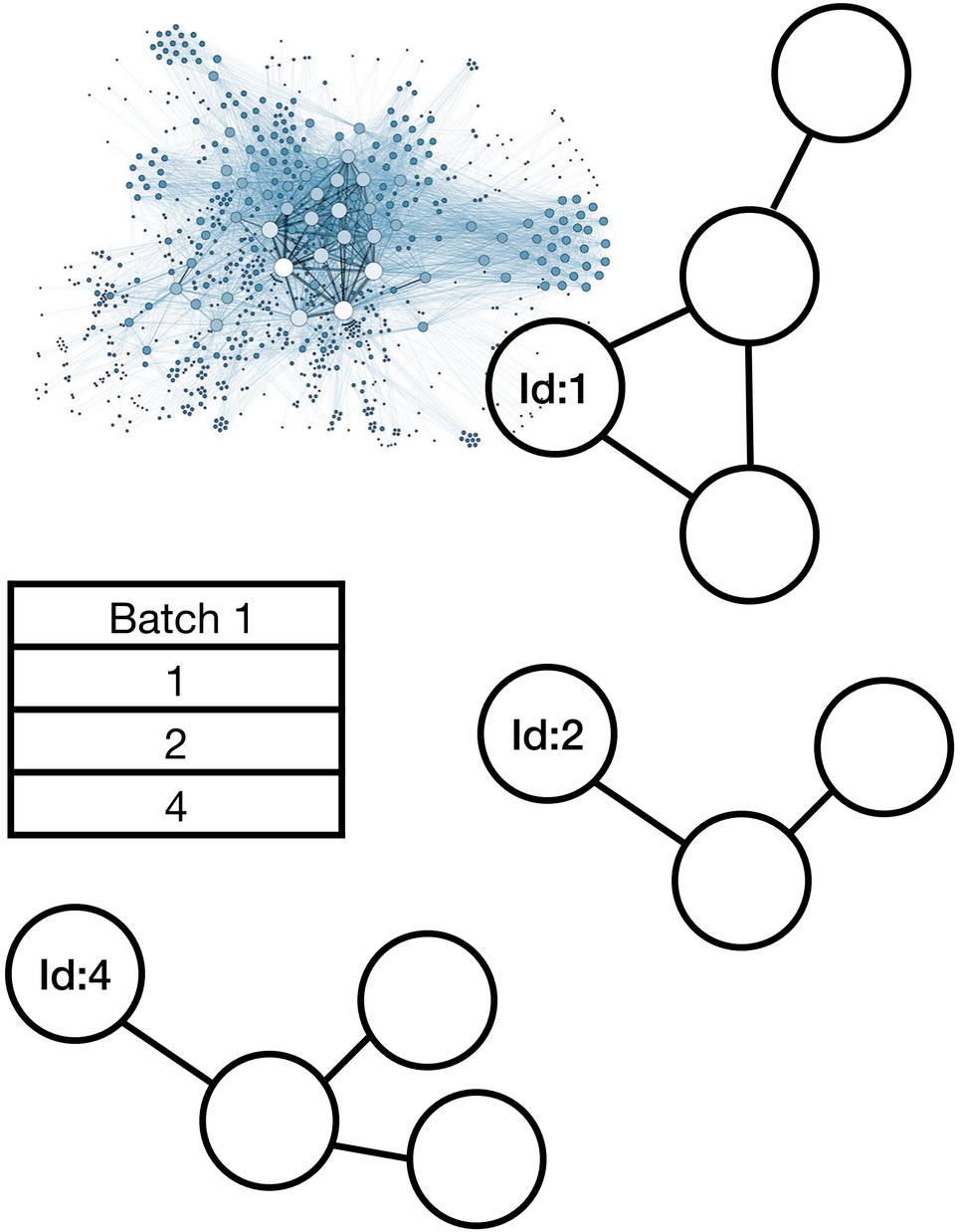
Batch 1
1
2
4

Batch 2
5
23
66

...

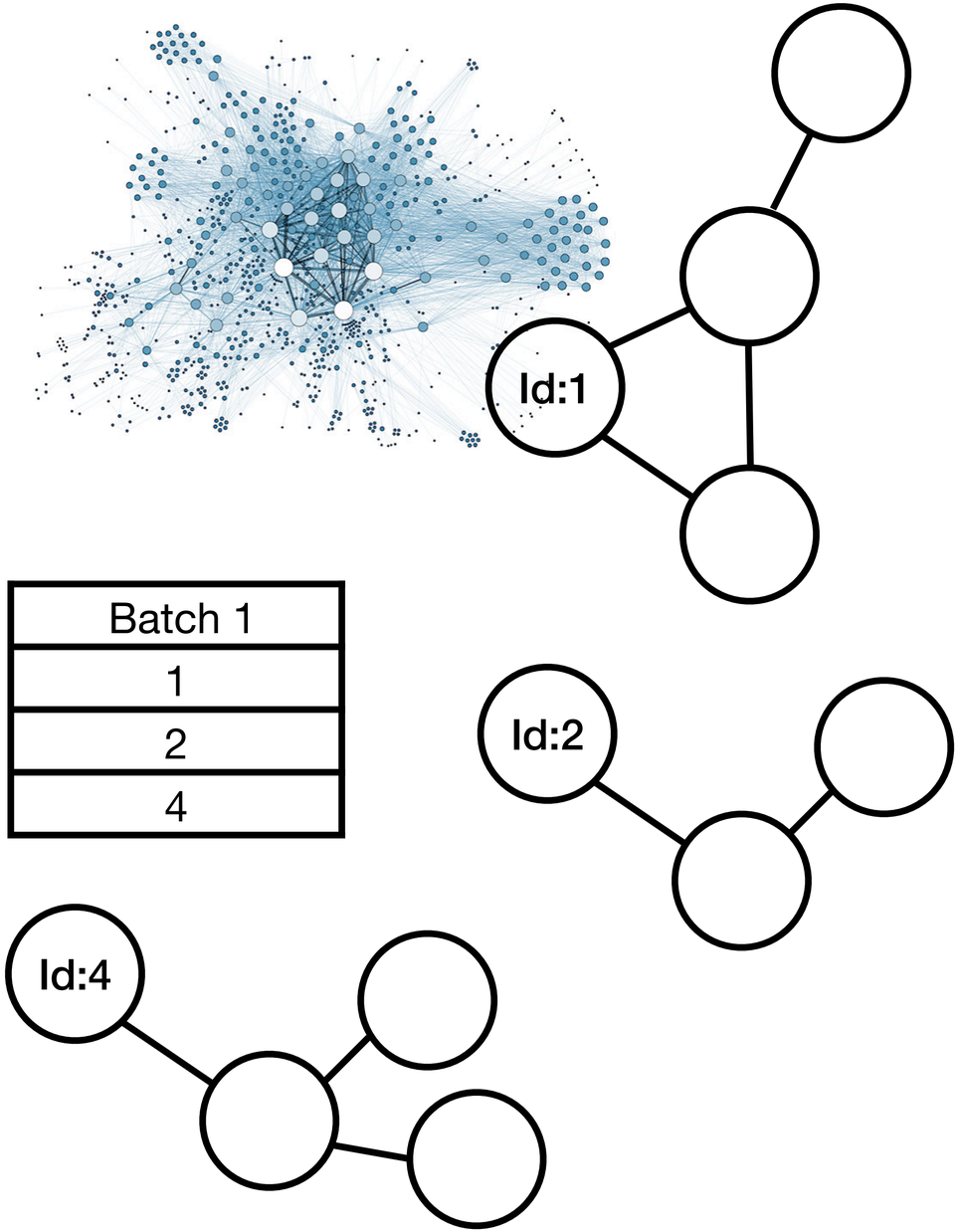
- Batch the set of nodes for training.

Training MPNN layers (supervised case) + batching (Hamilton et al. 2017)



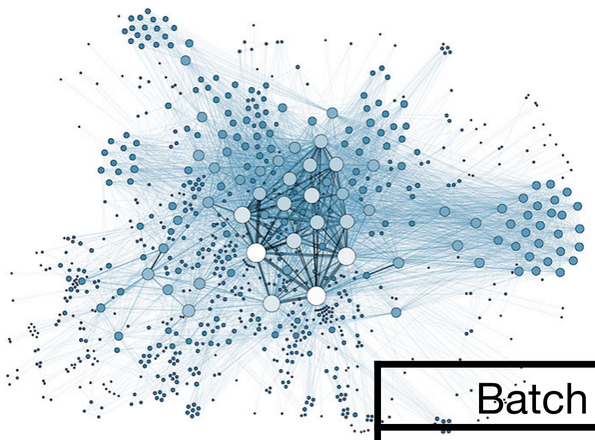
- Batch the set of nodes for training.
- For each batch, compute the neighbourhood (up to a distance according to the number of layers). May also limit to a sample of the set of neighbours at each step.

Training MPNN layers (supervised case) + batching (Hamilton et al. 2017)



- Batch the set of nodes for training.
- For each batch, compute the neighbourhood (up to a distance according to the number of layers). May also limit to a sample of the set of neighbours at each step.
- All these neighbourhoods constitute the graph we use for learning (in the batch).

Training MPNN layers (supervised case) + batching (Hamilton et al. 2017)



Training	Value
1	1
2	0
4	0
...	...
1000020	1
1000022	1

Batch 1
1
2
4

Batch 2
5
23
66

...

- Batch the set of nodes for training.
- For each batch, compute the neighbourhood (up to a distance according to the number of layers). May also limit to a sample of the set of neighbours at each step.
- All these neighbourhoods constitute the graph we use for learning.
- Again, we only compare training nodes, even though we get embeddings for much more nodes

Training MPNN layers (supervised case) + batching (Hamilton et al. 2017)

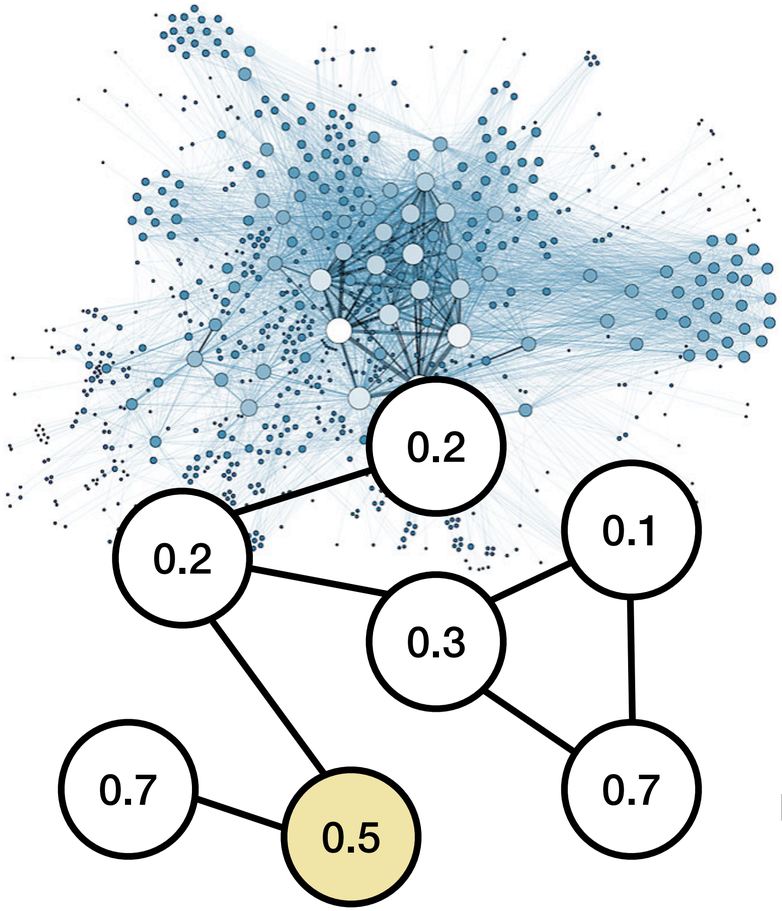
Batch 1	Value
1	1
2	0
4	0

v/s

Training	Value
1	0.84369
2	0.11241
4	0.12459

- Batch the set of nodes for training.
- For each batch, compute the neighbourhood (up to a distance according to the number of layers). May also limit to a sample of the set of neighbours at each step.
- All these neighbourhoods constitute the graph we use for learning.
- Again, we only compare training nodes, even though we get embeddings for much more nodes

Training MPNN layers (self - supervised case)



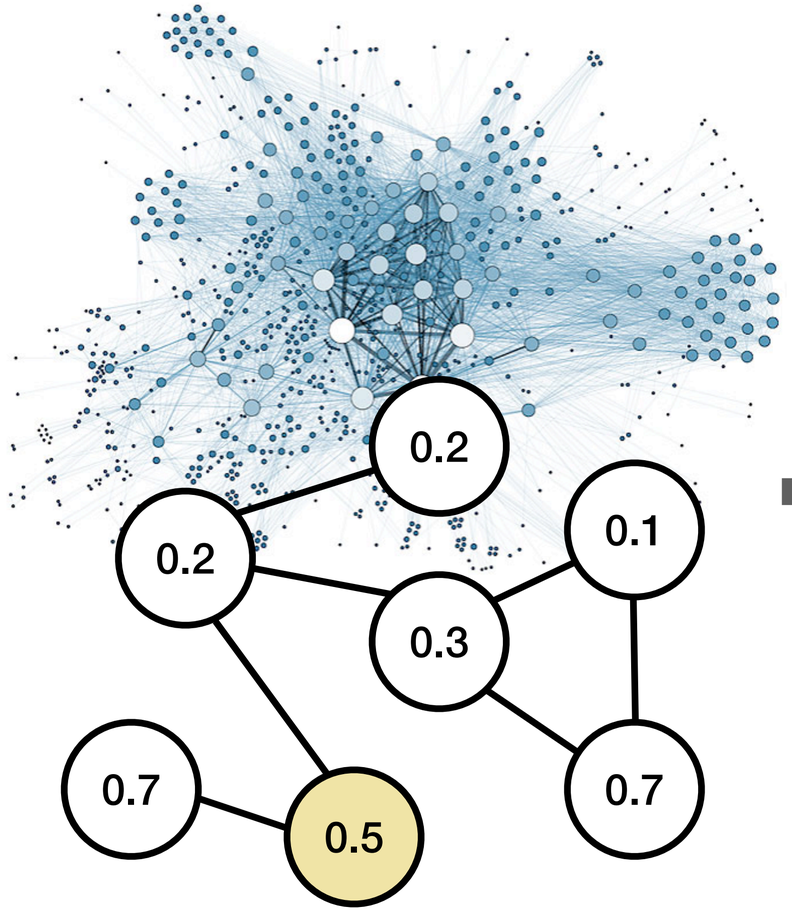
Forward Propagation

- Nodes start with features
- Layer i: all nodes update according to embeddings layer (i-1)
- Final layer: self supervised task, e.g. guessing whether there is an edge between two nodes.

Prediction

Training MPNN layers (self - supervised case)

- Final layer: self supervised task, e.g. guessing whether there is an edge between two nodes.



0.22452
0.84369
0.11241
0.12459
0.72629
0.78121
0.56291

0.22452
0.84369
0.11241
0.12459
0.72629
0.78121
0.56291

edge / no edge

GraphSAGE, Hamilton et al. 2017

MPNN layers to classify graphs

Training	Value
	1
	0
…	…
…	…
	
	

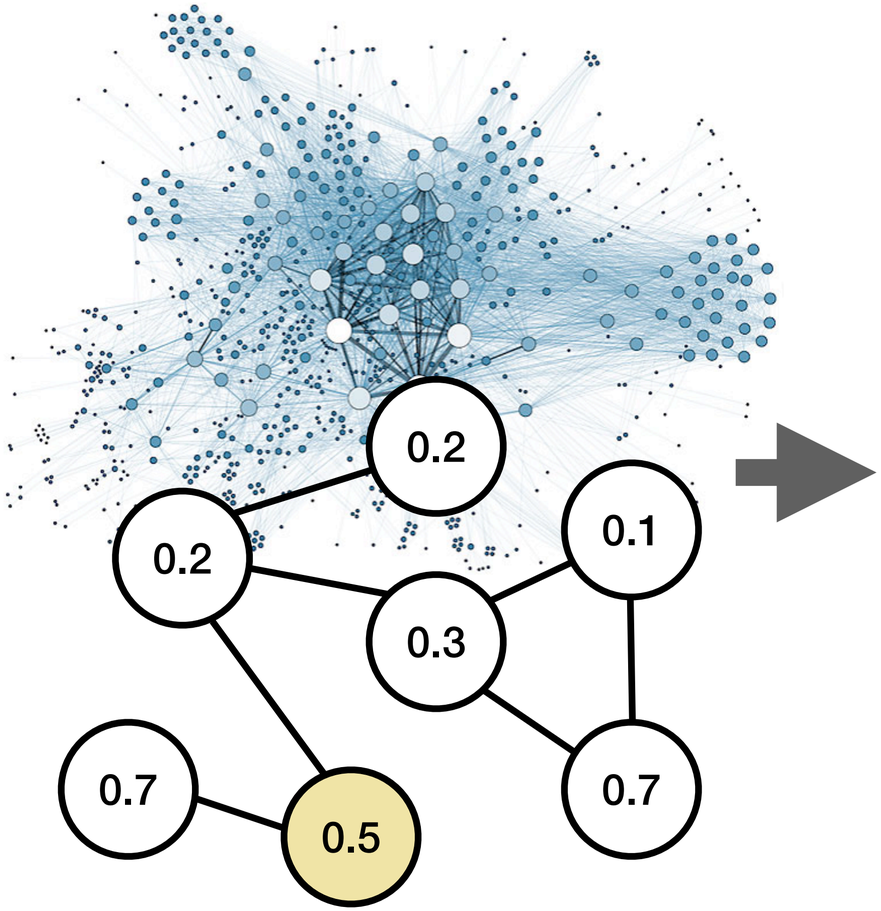
MPNN layers to classify graphs

Training	Value
	1
	0
…	…
…	…
	
	

- Only change: aggregator function at the end, takes embeddings of each node and produces a single graph embedding.

MPNN layers to classify graphs

- Only change: aggregator function at the end, takes embeddings of each node and produces a single graph embedding.



$$\sigma\left(\text{Aggregate}\left(\begin{array}{c} 0.22452 \\ 0.84369 \\ 0.11241 \\ 0.12459 \\ 0.72629 \\ 0.78121 \\ 0.56291 \\ \ldots \end{array}\right)\right) = \boxed{0.42851}$$

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

How powerful are MPNNs?

Message Passing Neural Networks

MPNNs: set of subsequent MPNN layers.

Relatively quick to train (in terms of edges in the graph)

Can be learned effectively

Implementations available

How powerful are MPNNs?

Message Passing Neural Networks

MPNNs: set of subsequent MPNN layers.

Relatively quick to train (in terms of edges in the graph)

Can be learned effectively

Implementations available

But are they useful?

Separation Power

Separation Power

Graph version

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

(Over any GNN)

Node version

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

(Over any GNN)

Separation Power

Equivalence relation given by graphs/nodes for which each GNN computes the same embedding

Graph version

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

(Over any GNN)

Node version

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

(Over any GNN)

Separation Power

Graph version

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

(Over any GNN)

Node version

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

(Over any GNN)

Smaller relation $\longrightarrow$ **More separation power**

Separation Power

Equivalence relation given by graphs/nodes for which each GNN computes the same embedding

Graph version

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

(Over any GNN)

Node version

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

(Over any GNN)

Smaller relation $\longrightarrow$ **More separation power**

Separation Power

Graph version

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

(Over any GNN)

Smaller relation $\longrightarrow$ **More separation power**

$\rho(GNN_1) \subseteq \rho(GNN_2)$: GNN_1 distinguishes more graphs than GNN_2

Separation Power

Equivalence relation given by graphs/nodes for which each GNN computes the same embedding

Graph version

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

(Over any GNN)

Smaller relation $\longrightarrow$ **More separation power**

$\rho(GNN_1) \subseteq \rho(GNN_2)$: GNN_1 distinguishes more graphs than GNN_2

Separation Power

We are interested in separation power of GNNs architectures.
Which architectures distinguish more pairs of graphs.

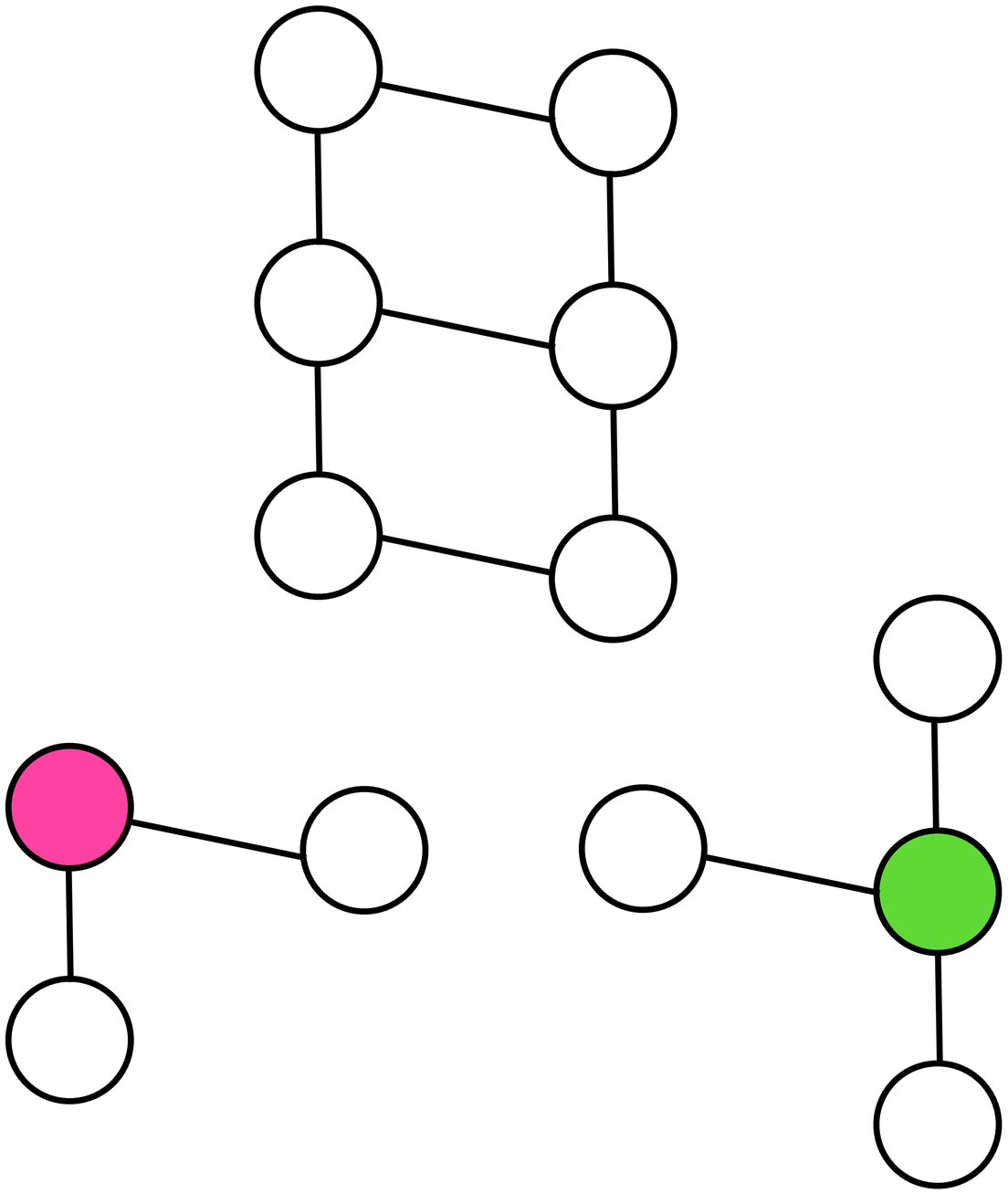
Separation Power

Equivalence relation given by graphs/nodes for which
each GNN computes the same embedding

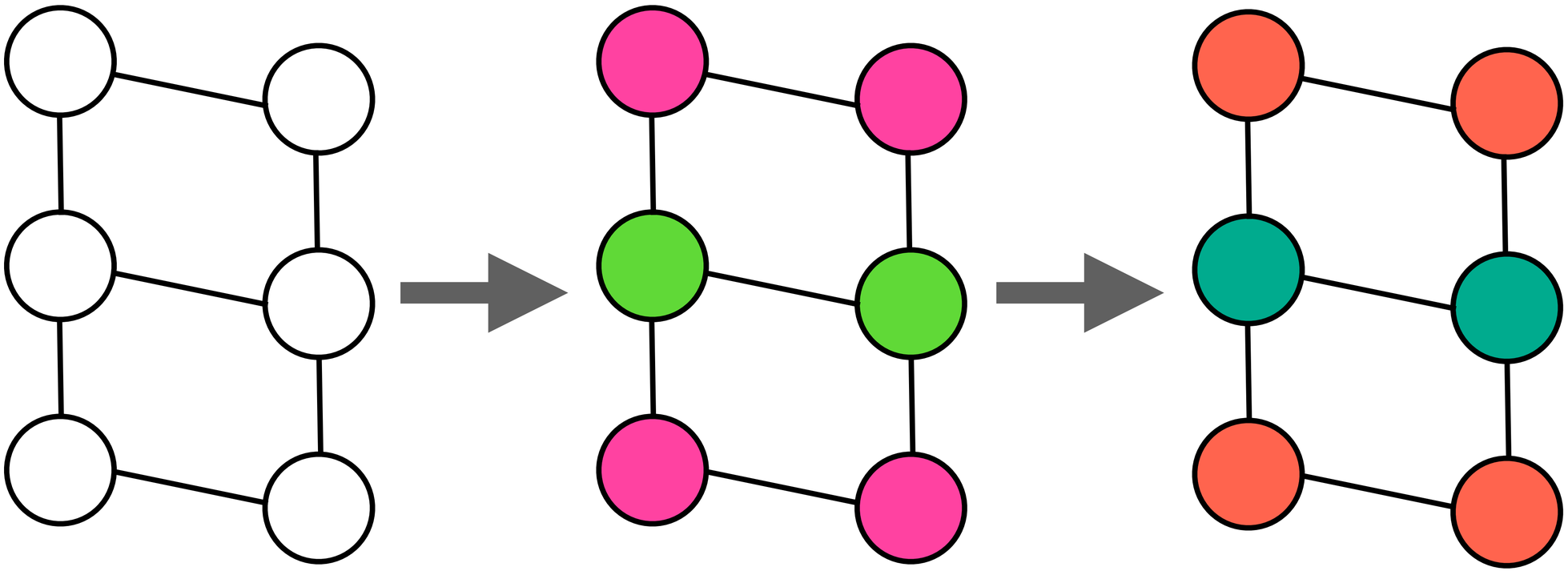
We are interested in separation power of GNNs architectures.
Which architectures distinguish more pairs of graphs.

Weisfeiler-Lehman test (Colour refinement)

- Initial colouring based on node features
- Update colour based on multiset of colours of self and neighbours
- Stop when no new vertex is separated



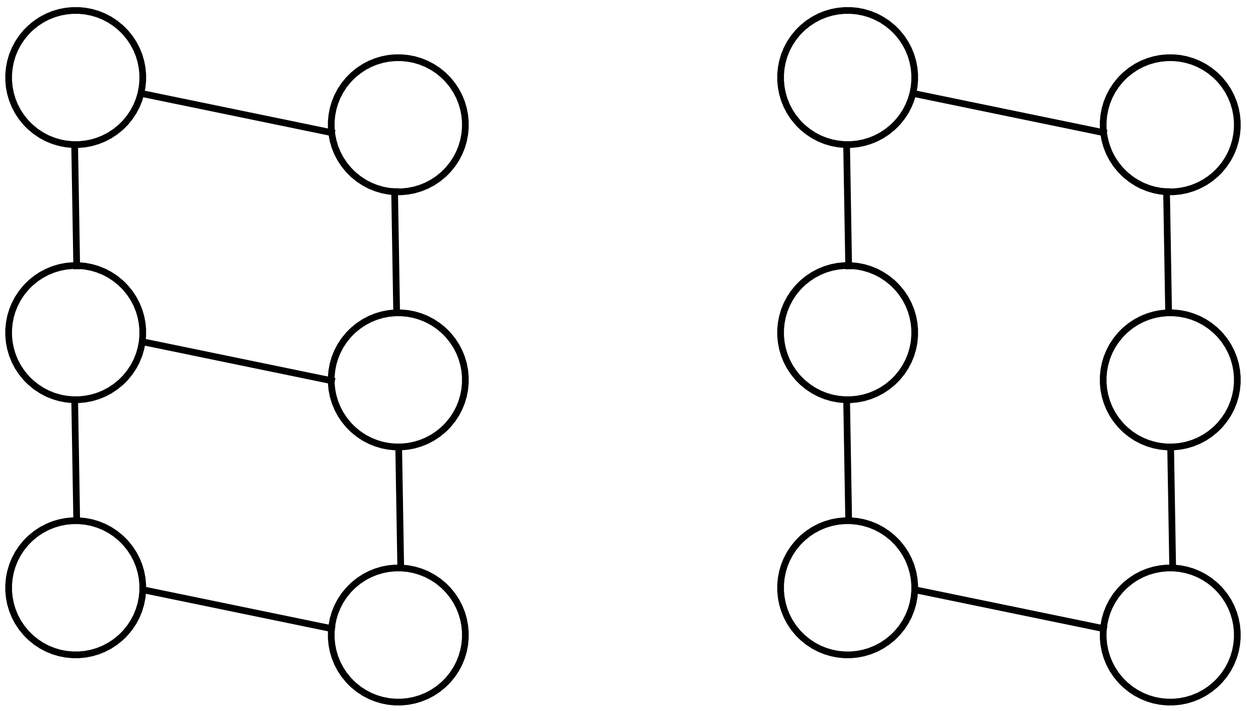
Weisfeiler-Lehman test
(Colour refinement)



No new node is separated

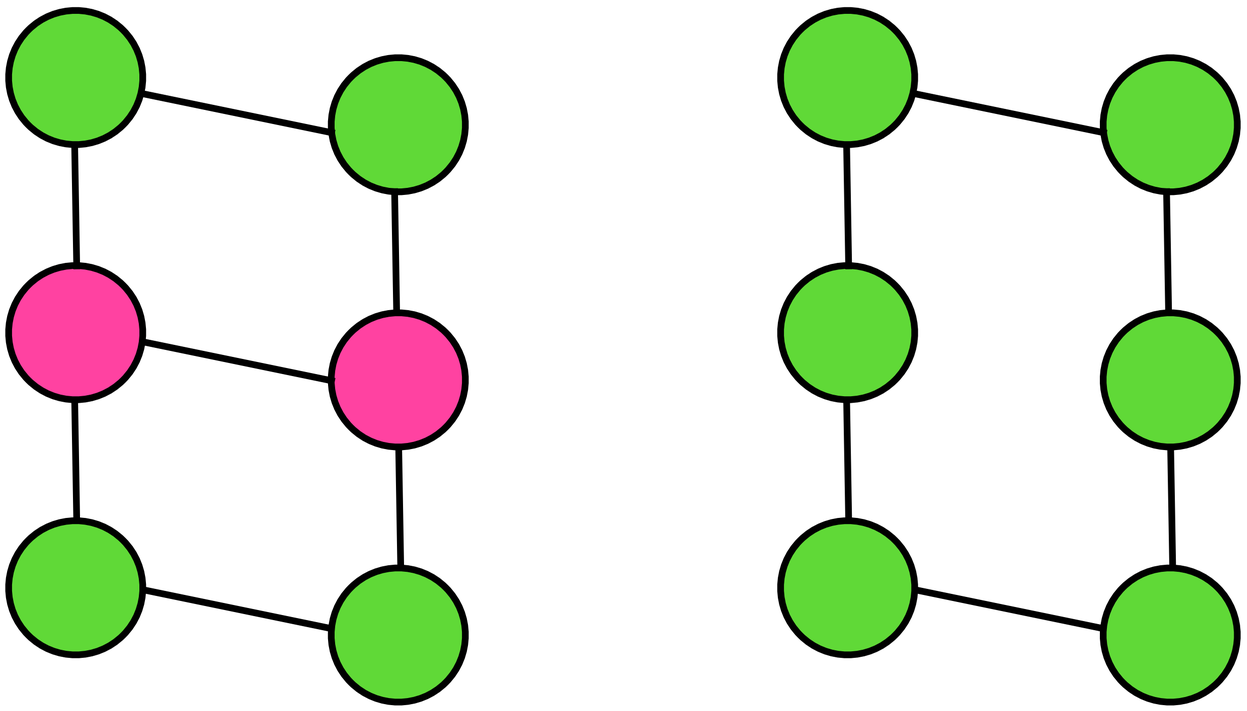
(Equivalence relation of colours did not change)

Colour refinement for graph embeddings



Colours of vertices can be understood as embeddings

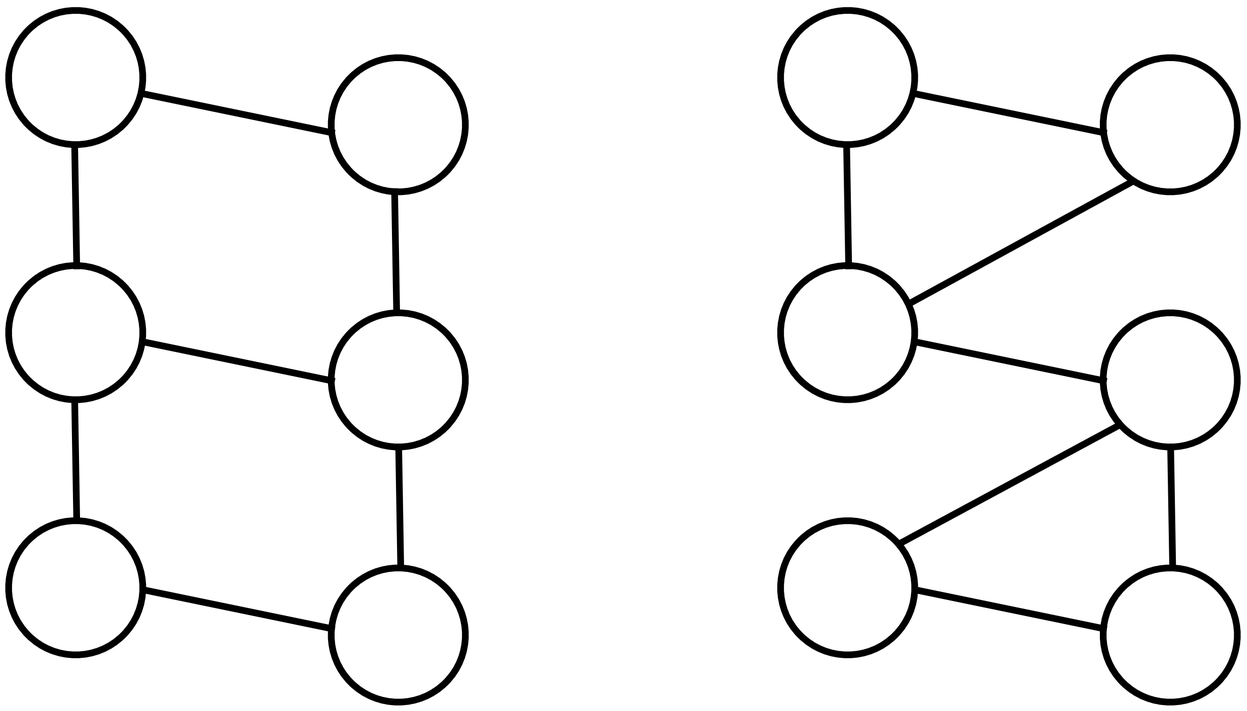
Colour refinement for graph embeddings



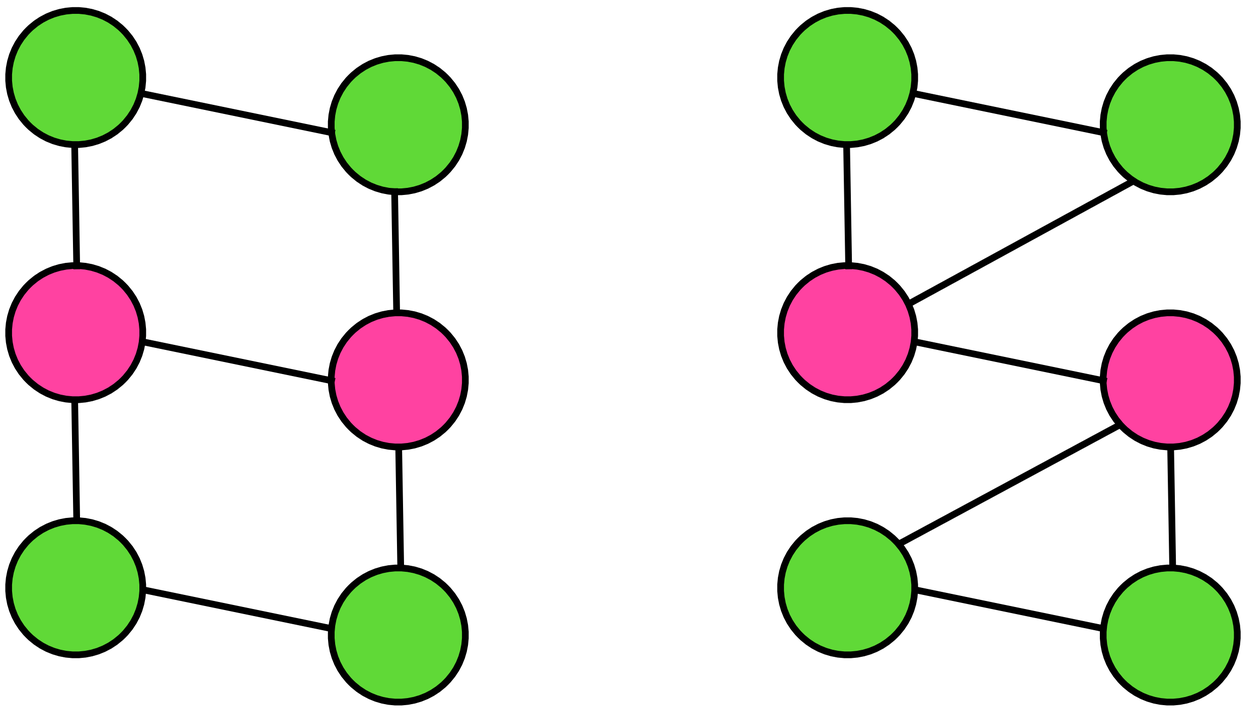
Different graphs:

Multiset of colours of vertices is different

Colour refinement for graph embeddings



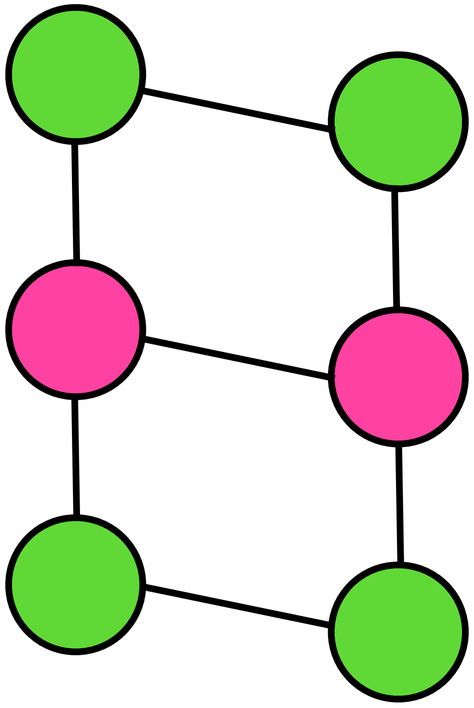
Colour refinement for graph embeddings



Same graph embedding

These graphs are not separated

Colour refinement for node embeddings



Nodes of the same colour are not separated

Separation Power of MPNNs

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

The separation power of MPNNs is bounded by:

- **Colour refinement, for node embeddings, and**
- **1-dimensional WL algorithm, for graph embeddings**

Separation Power of MPNNs

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

The separation power of MPNNs is bounded by:

- **Colour refinement, for node embeddings, and**
- **1-dimensional WL algorithm, for graph embeddings**

Similar test as CR

Separation Power of MPNNs

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

The separation power of MPNNs is bounded by:

- **Colour refinement, for node embeddings, and**
- **1-dimensional WL algorithm, for graph embeddings**

Take an arbitrary MPNN M

Then

$$\rho(CR) \subseteq \rho(M)$$

Separation Power of MPNNs

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

The separation power of MPNNs is bounded by:

- **Colour refinement, for node embeddings, and**
- **1-dimensional WL algorithm, for graph embeddings**

If Colour Refinement cannot tell two nodes are different
Then MPNNs cannot do it either

Separation Power of MPNNs

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

The separation power of MPNNs is bounded by:

- **Colour refinement, for node embeddings, and**
- **1-dimensional WL algorithm, for graph embeddings**

If 1-dim WL cannot tell two graphs are different
Then MPNNs cannot do it either

Separation Power of MPNNs

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

The separation power of MPNNs is bounded by:

- **Colour refinement, for node embeddings**

Proof is by induction. If after some layer the above holds, then the result of an MPNN layer will continue bounded on colour refinement.

Separation Power of MPNNs

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

The separation power of MPNNs is bounded by:

- **Colour refinement, for node embeddings, and**
- **1-dimensional WL algorithm, for graph embeddings**

Further, there are MPNN architectures that approximate the results of these tests (up to any precision).

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

But what can MPNNs compute?

1. in terms of logical queries
2. In terms of arbitrary functions

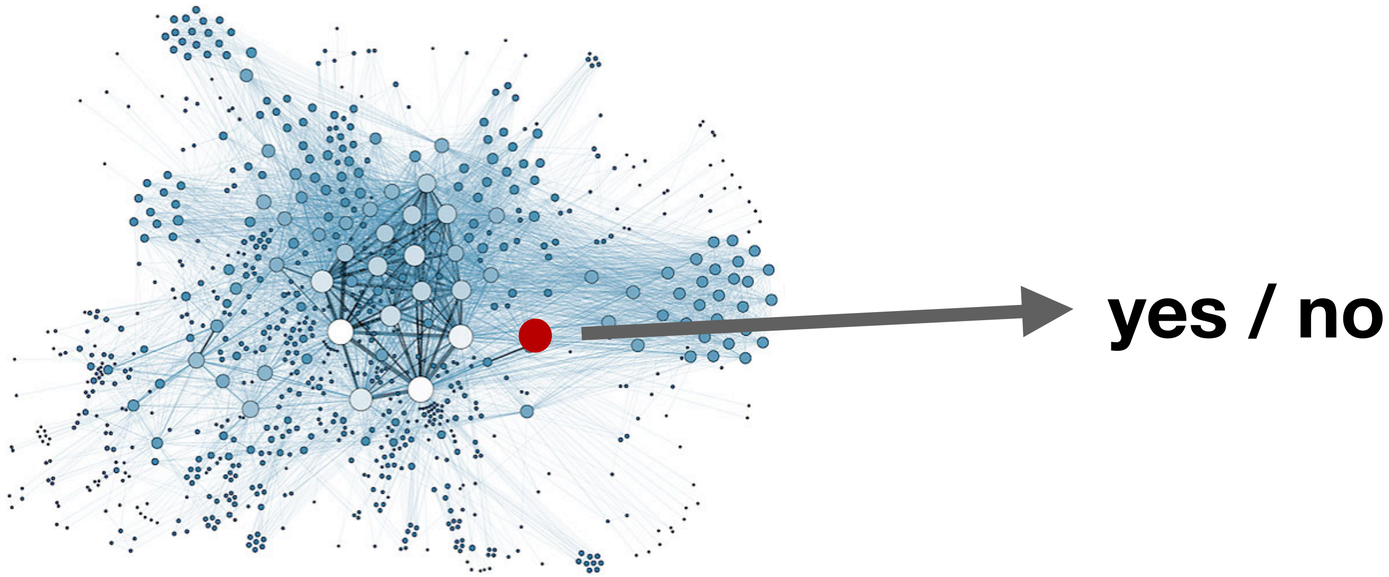
But what can MPNNs compute?

Lets start with logical functions

yes / no

But what can MPNNs compute?

Lets start with logical functions



But what can MPNNs compute?

Take the set of all functions from graphs to {0,1},

Take the set of all functions from graphs and one of its nodes to {0,1}

Which ones can be captured by MPNNs?

Theorem (Cai et al.):

The following are equivalent in any graph:

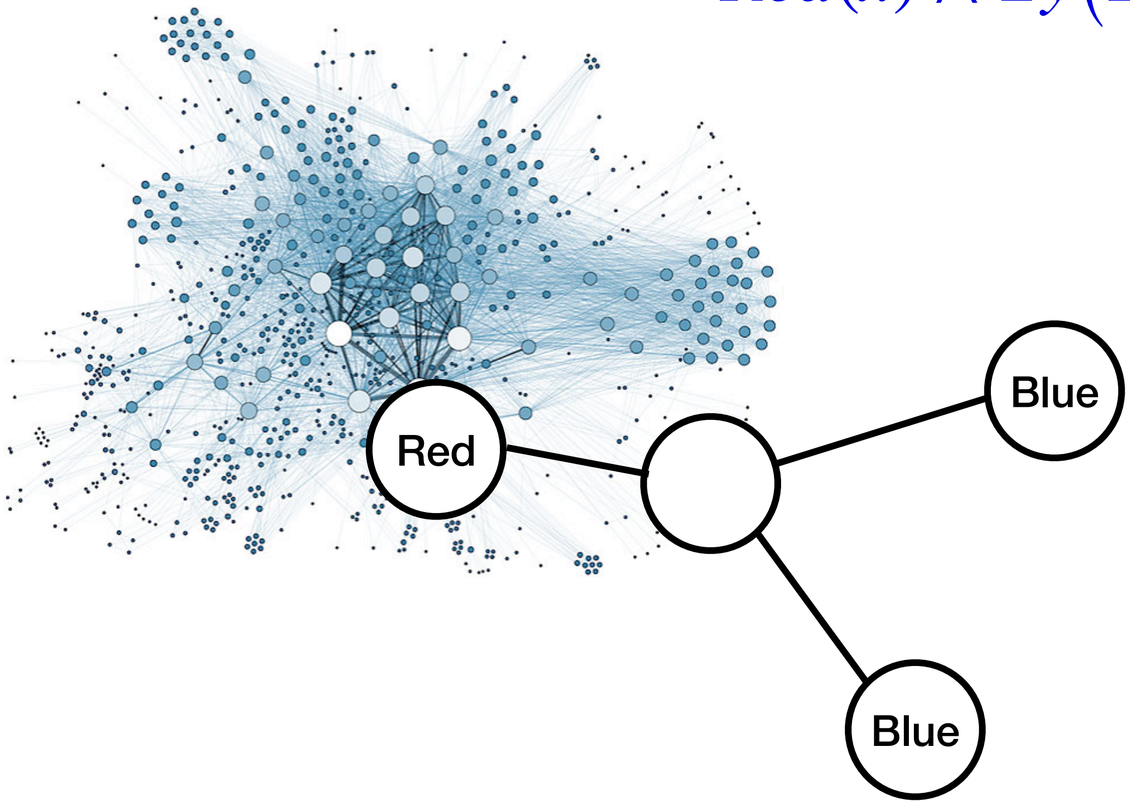
- **Two nodes in a graph have the same Colour Refinement**
- **Two nodes in a graph are equivalent w.r.t. FOC2**

FOC2: First order logic with 2 variables, but with quantifiers $\exists^{\geq N}$

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

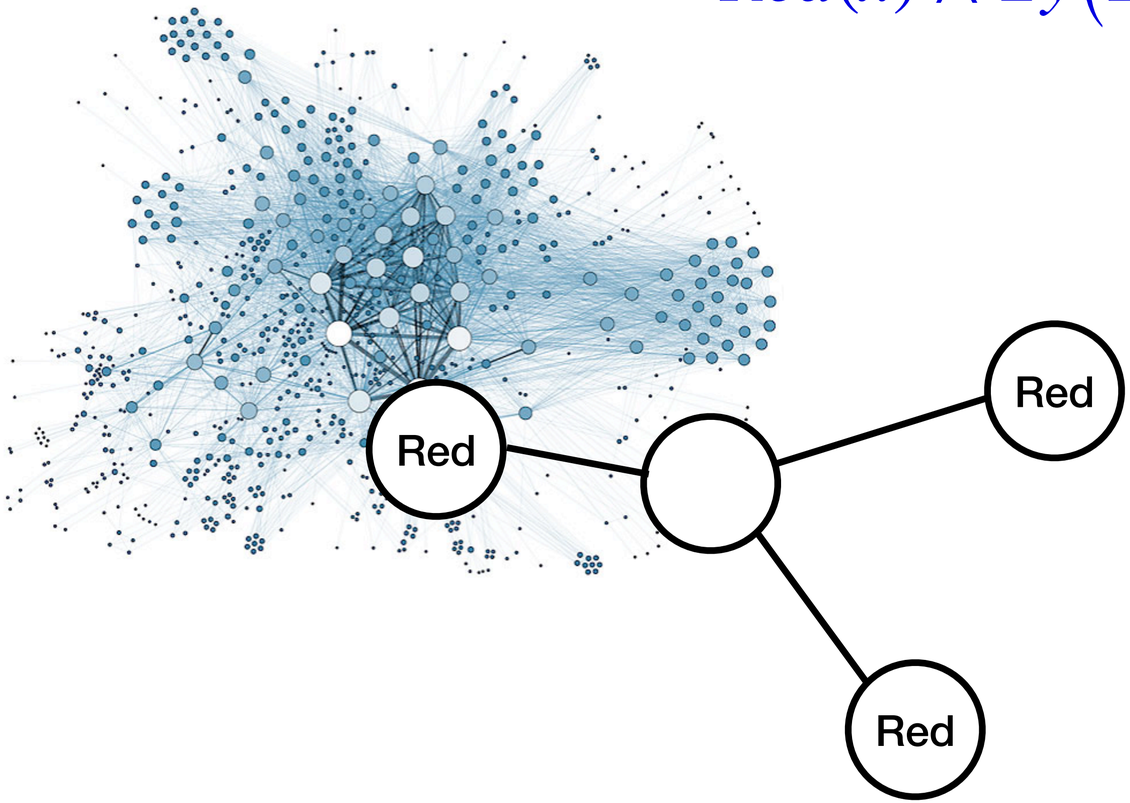
FOC2, tree unravellings (and counting)

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



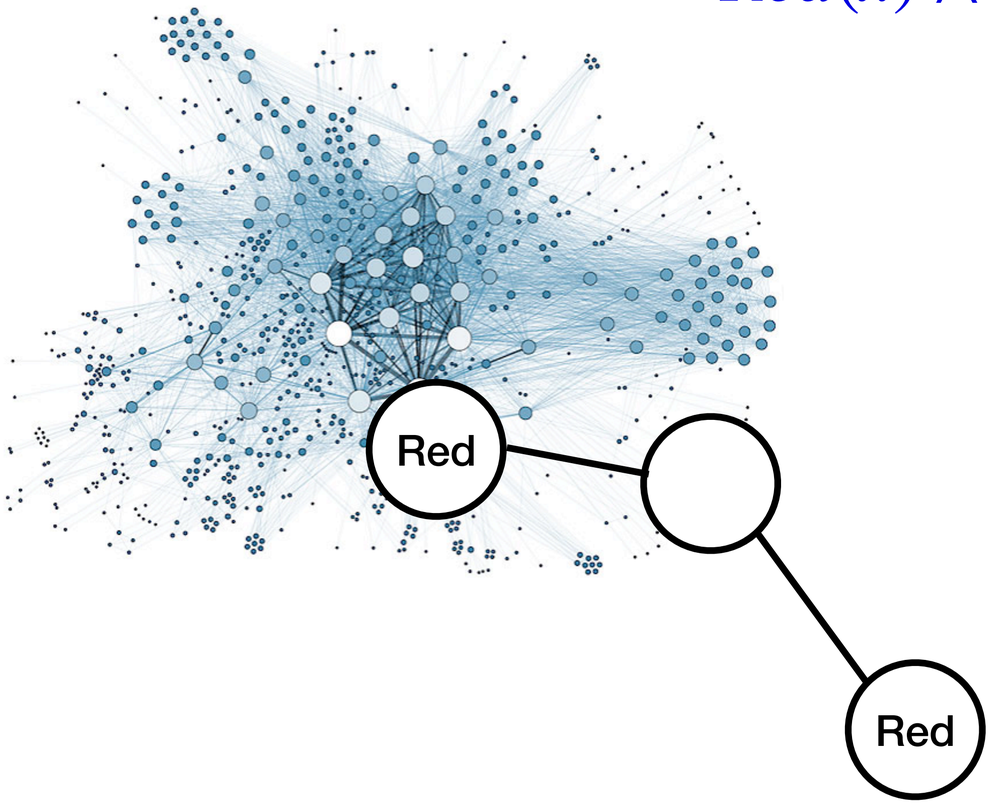
FOC2, tree unravellings (and counting)

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



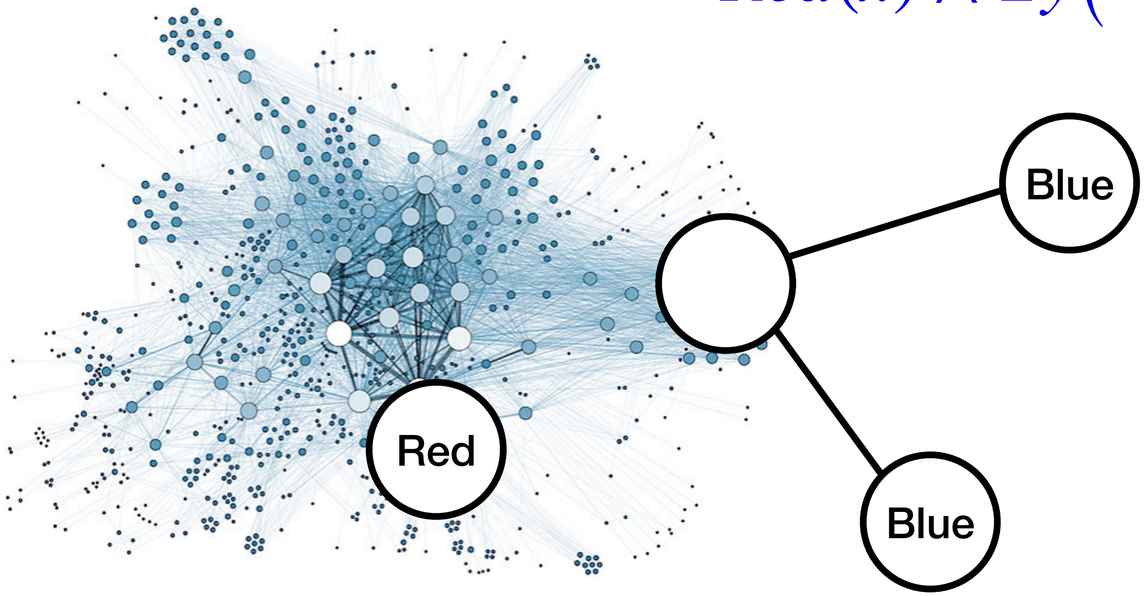
FOC2, tree unravellings (and counting)

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



FOC2, tree unravellings (and counting)(and more!)

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



Theorem (Cai et al.):

The following are equivalent in any graph:

- **Two nodes in a graph have the same Colour Refinement**
- **Two nodes in a graph are equivalent w.r.t. FOC2**

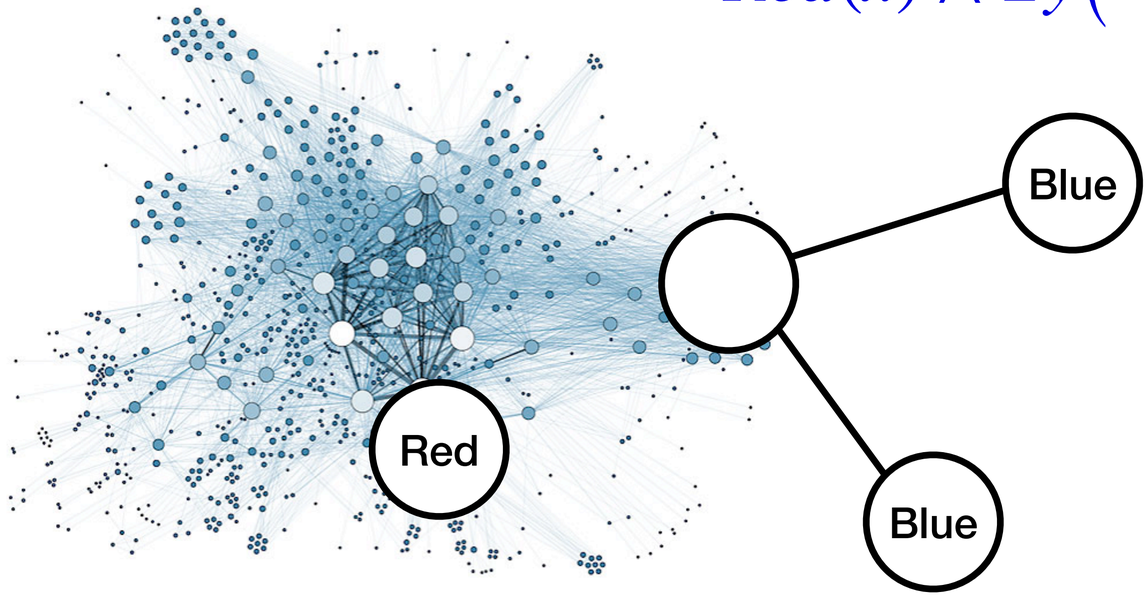
Take the set of all functions from graphs to {0,1},

Take the set of all functions from graphs and one of its nodes to {0,1}

Which ones can be captured by MPNNs?

FOC2, tree unravellings (and counting)(and more!)

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



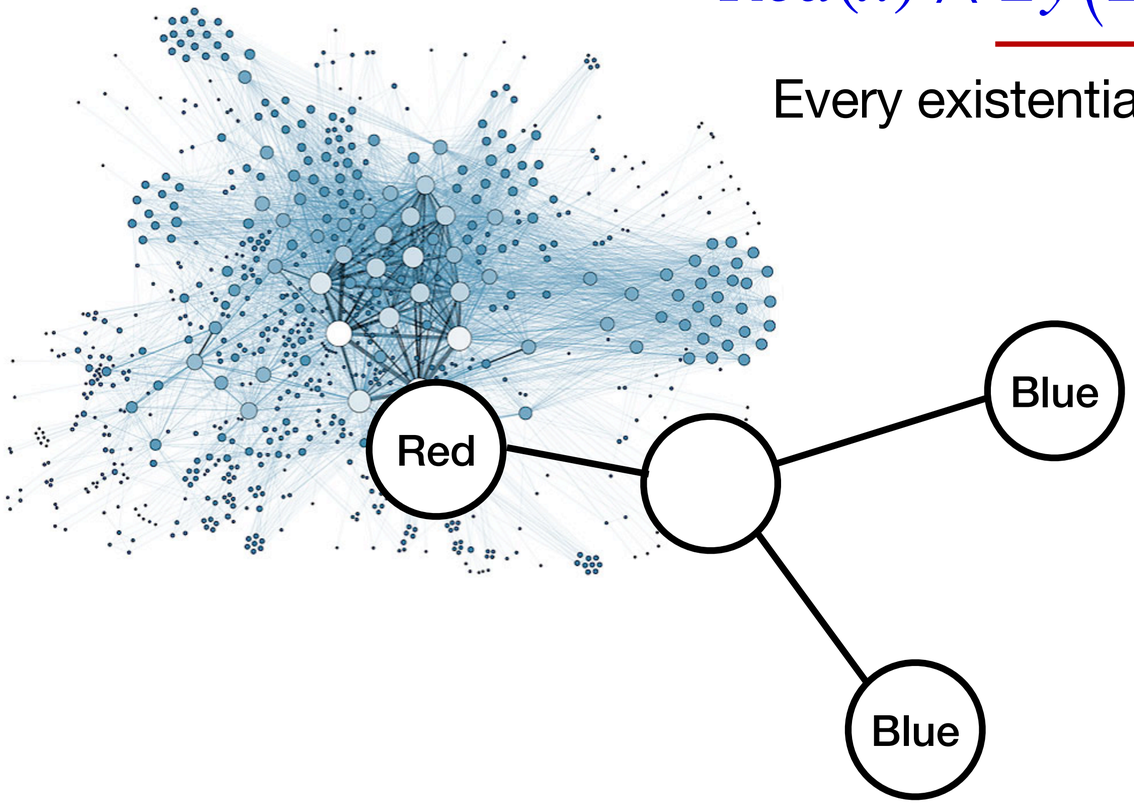
This one cannot be captured by an MPNN!

Can we see why?

Guarded FOC2 = tree unravellings (and counting)

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

Every existential is guarded by Edge



Theorem (Barceló et al.):

A binary function from (graph, nodes) to {0,1} can be expressed as an MPNN if and only if it can be expressed in guarded FOC2.

Theorem (Barceló et al.):

A binary function from (graph, nodes) to {0,1} can be expressed as an MPNN if and only if it can be expressed in guarded FOC2.

Full FOC2 can be captured by enhancing MPNNs with a single global aggregation layer at the end.

But what can MPNNs compute?

In terms of binary functions, they resemble **guarded FOC2**.

But **MPNNs** can do arbitrary functions!

Function Approximation

Consider a set F of functions.

Closure of F contains functions h such that:

- **There is a sequence of functions from F**
- **Each new function gets closer and closer to h**

Idea: using F we can approximate anything in its closure

But what can MPNNs compute?

Theorem:

MPNN architectures can approximate any function whose separation power is bounded by Colour Refinement

(separation power)

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

But what can MPNNs compute?

Take a function f.

Suppose that separation power of f is bounded by separation power of 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?

Take a function f.

Suppose that separation power of f is bounded by separation power of colour refinement.

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

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

Then f can be approximated by an MPNN-based architecture, up to any precision

(As long as certain mild conditions are satisfied)

So… MPNNs

Model is cheap.

So… MPNNs

Model is cheap.

Also, MPNN layers provide a nice, trainable way of capturing graph info (better than passing the adjacency matrix anyways!)

So… MPNNs

Model is cheap.

Also, MPNN layers provide a nice, trainable way of capturing graph info (better than passing the adjacency matrix anyways!)

But, not very expressive:

- cannot count triangles
- Misses long path information
- Self-supervised has further problems (see e.g. Rampášek et al.)

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

Beyond MPNNs

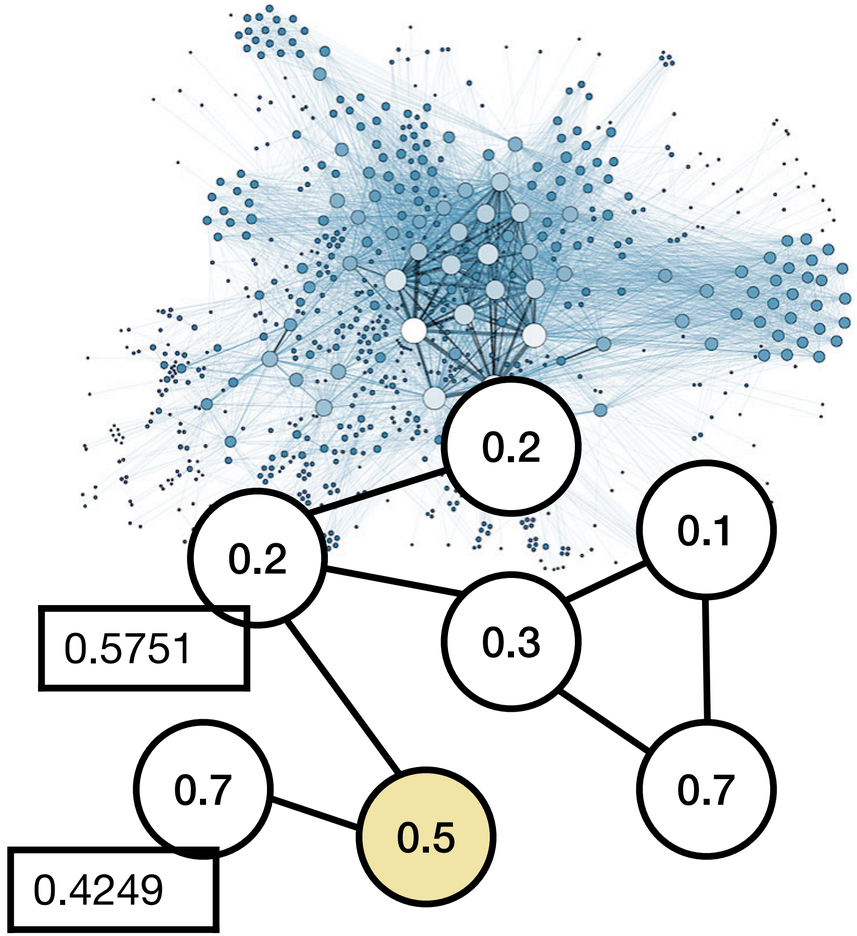
(Not comprehensive) survey on GNNs more expressive than MPNNs.

Goal is to understand architectural designs

Beyond MPNNs

1. Attention (Veličković et al. 2017)

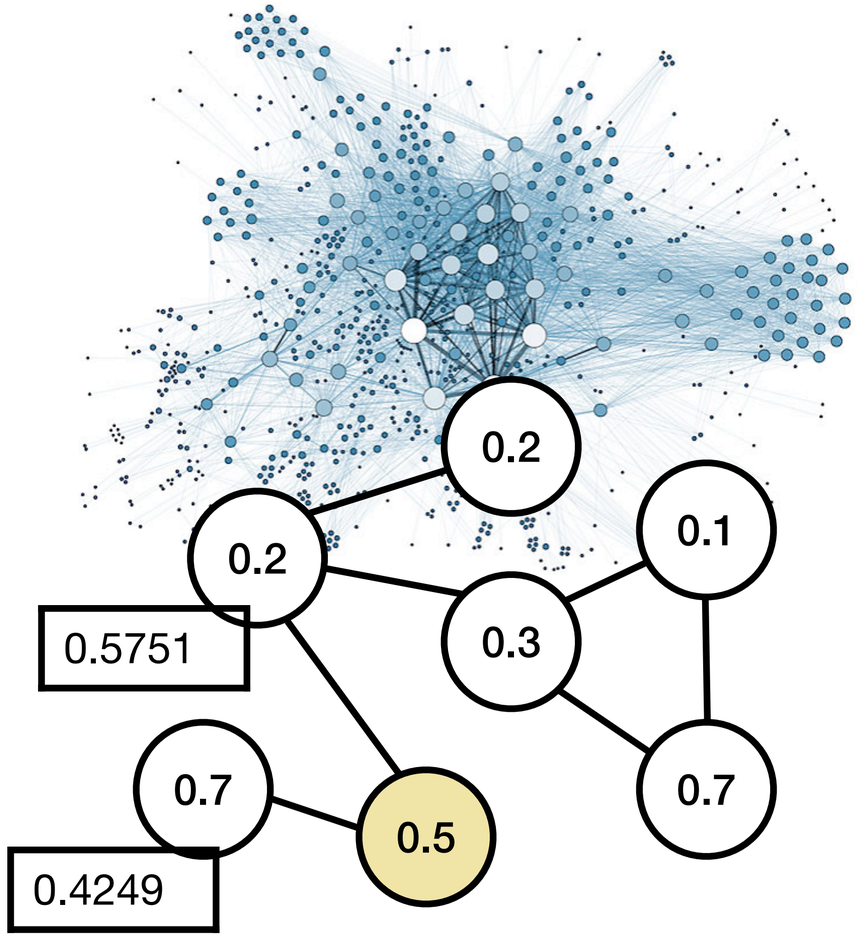
Attention weight applied to each neighbour before aggregation



Beyond MPNNs

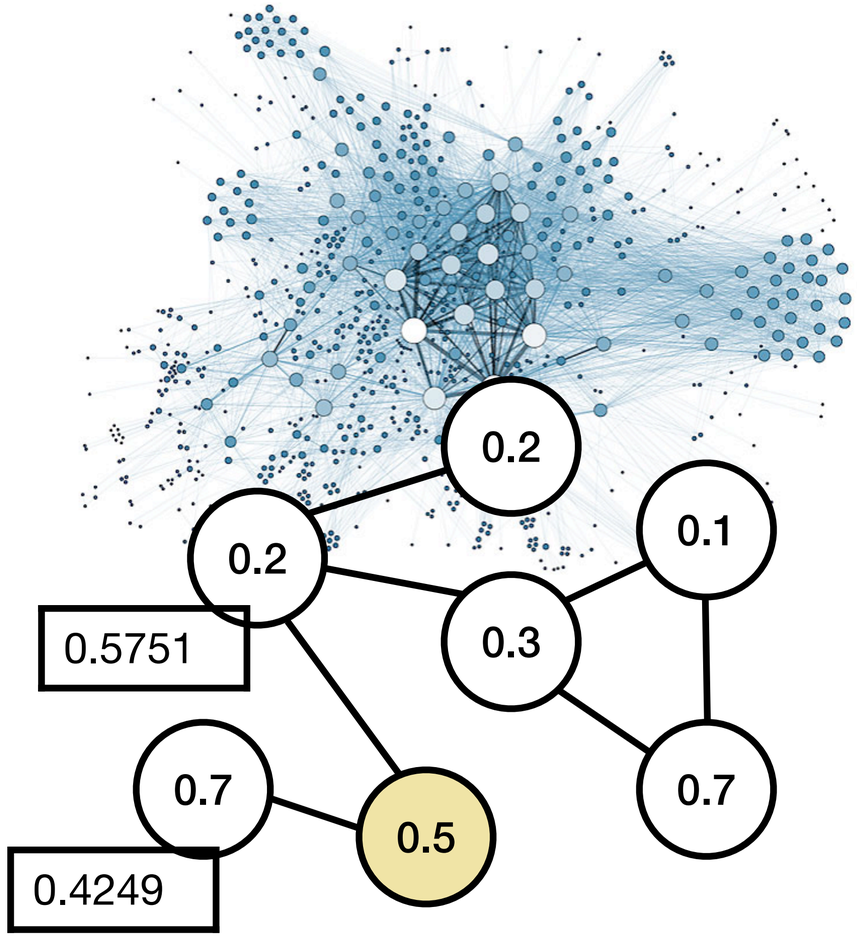
1. Attention (Veličković et al. 2017)

Attention weight applied to each neighbour before aggregation



Separation power bounded by Colour Refinement

Beyond MPNNs



2. Extra node/graph info

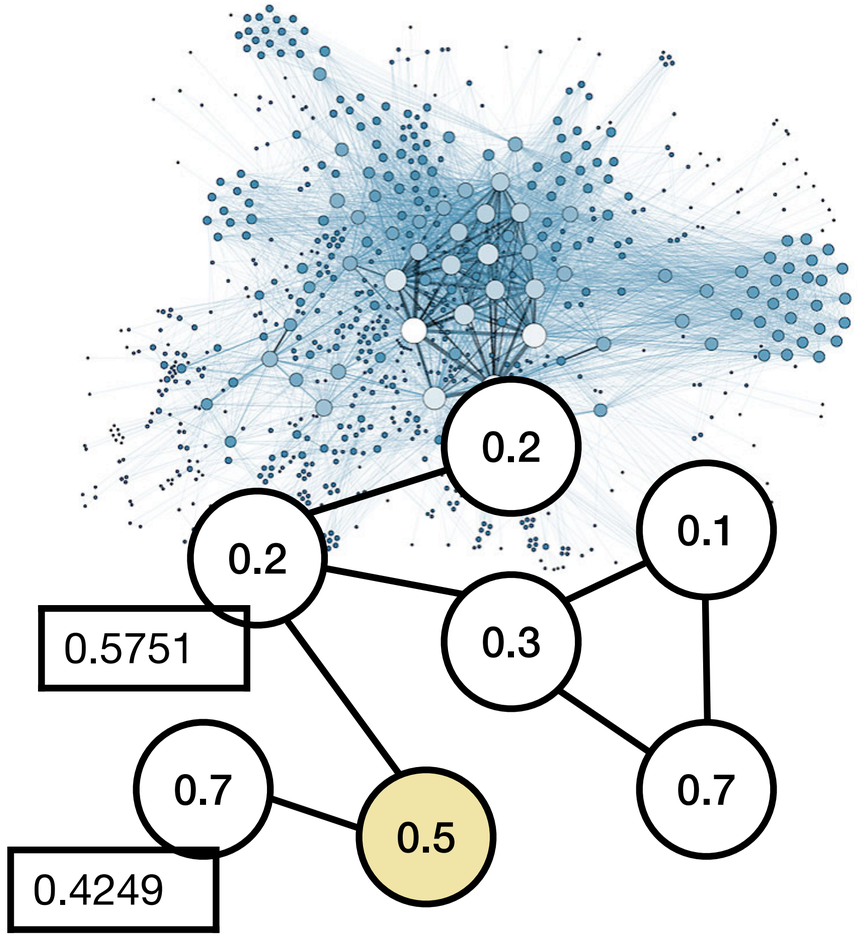
Subgraph info (e.g. Bouritsas et al. 2020, Barceló et al. 2021)

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)

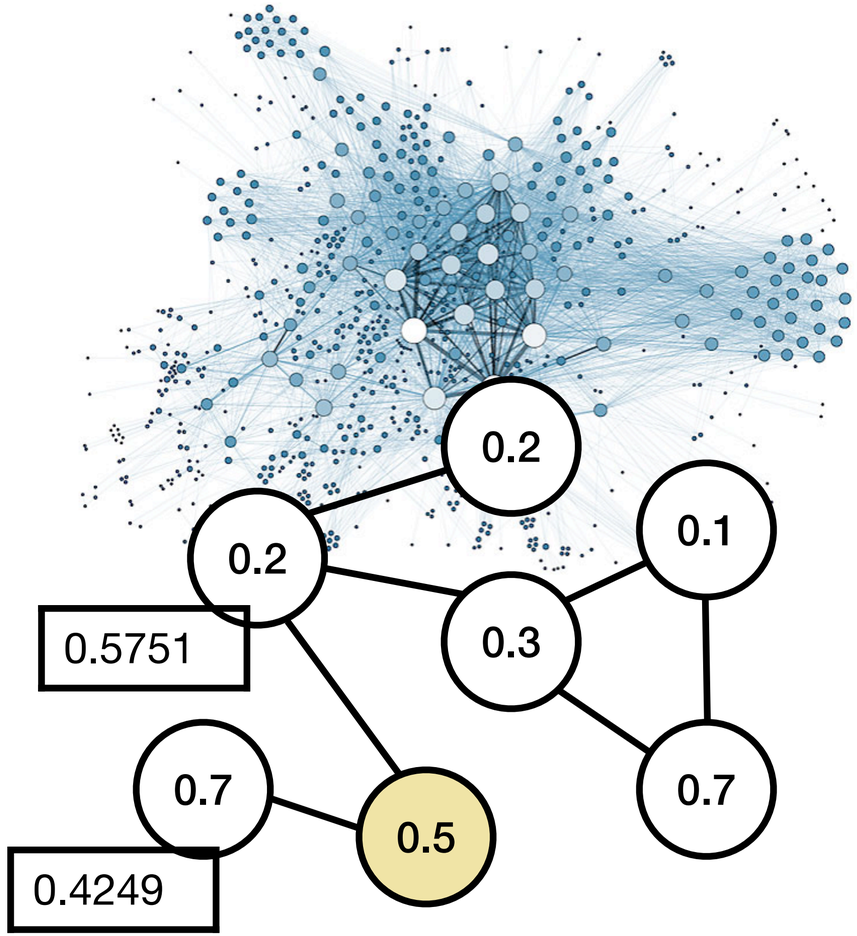
Spectral information (e.g. Kreuzer et al. 2021)

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

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

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

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

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

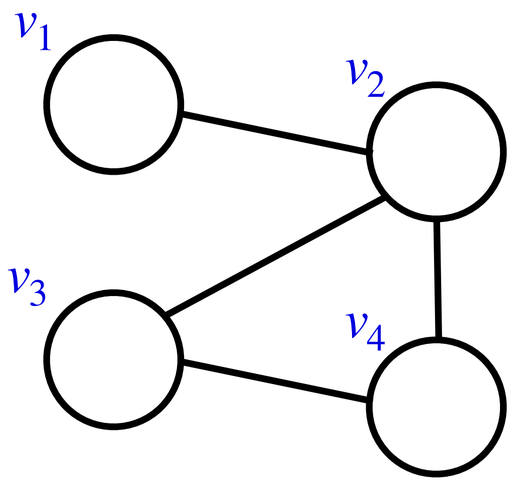
Edge Boolean test for edge between two vertices

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

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

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

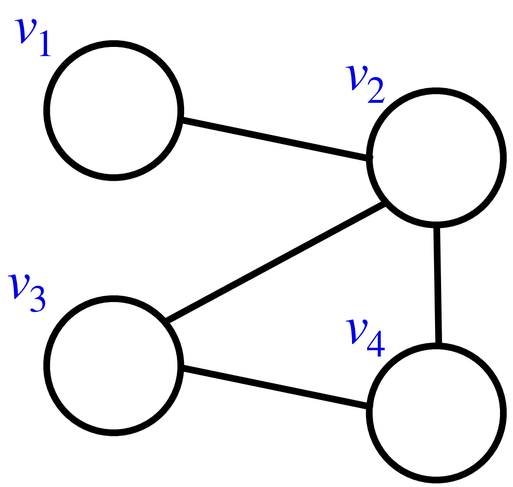
Aggregation



$$emb_{x_1 x_2}$$

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

Aggregation



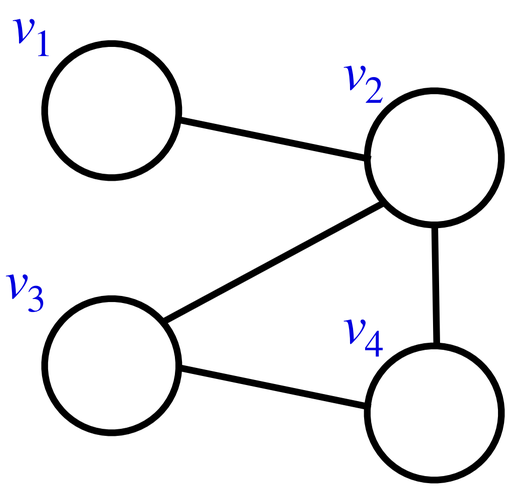
$emb_{x_1 x_2}$

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

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

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

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

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

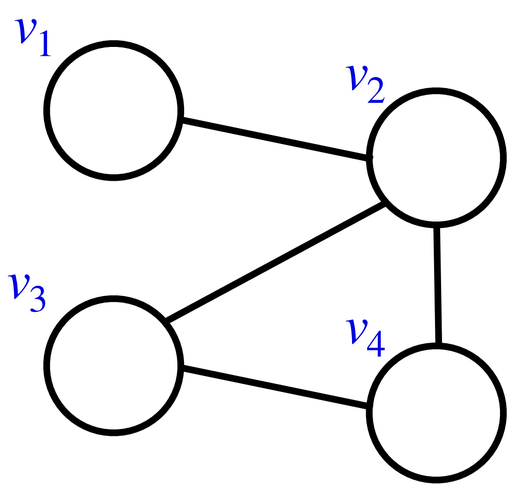
$(v_1) \mapsto 0.3$

$(v_2) \mapsto \ldots$

$\vdots$

over all pairs satisfying this condition

Aggregation



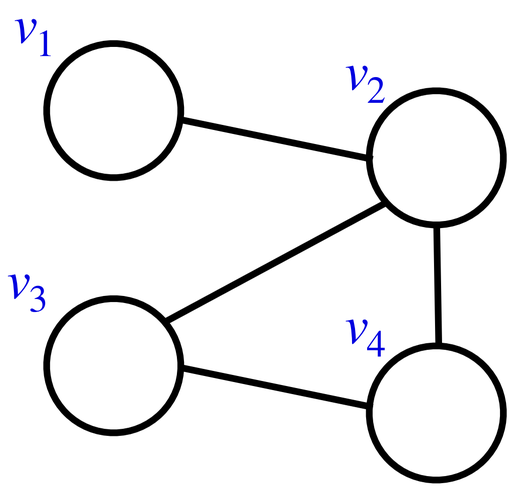
$emb_{x_1 x_2}$

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

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

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

Aggregation



$emb_{x_1 x_2}$

$(v_1, v_1) \mapsto 0.1$

$(v_1, v_2) \mapsto 0.5$

$(v_1, v_3) \mapsto 0.1$

$(v_1, v_4) \mapsto 0.5$

$\vdots$

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

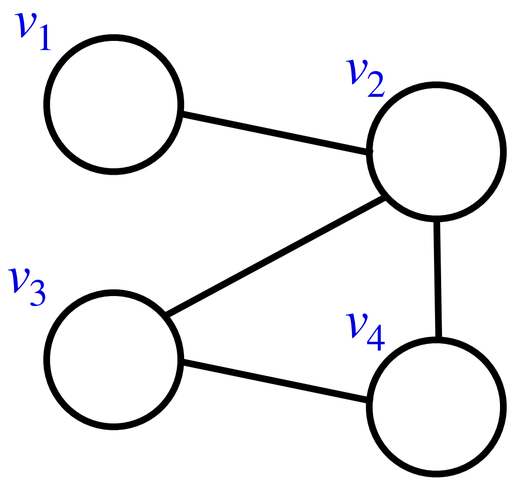
Only neighbours satisfy this

$(v_1) \mapsto 0.5$

$(v_2) \mapsto \ldots$

$\vdots$

Function application



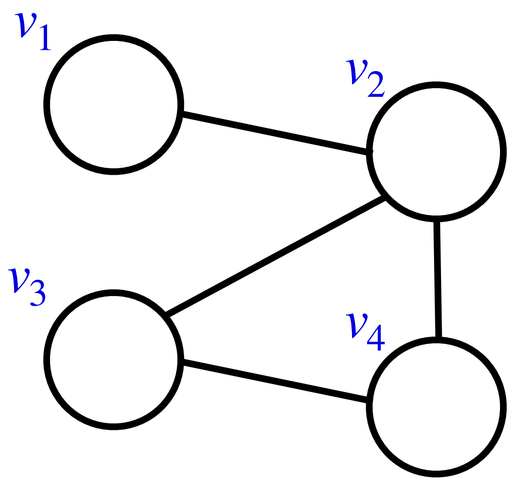
$emb^1_{x_1x_2}$

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

$emb^2_{x_2x_3}$

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

Function application



$emb^1_{x_1x_2}$

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

$\vdots$

$emb^2_{x_2x_3}$

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

$\vdots$

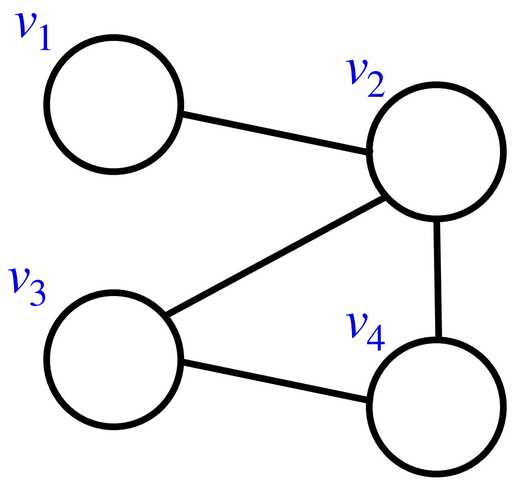
$MLP(emb^1_{x_1x_2}, emb^2_{x_2,x_3})$

$\vdots$

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

$\vdots$

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

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)

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

Examples.

2-Folklore GNNs (Maron et. al 2020)

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

Examples.

2-Folklore GNNs (Maron et. al 2020)

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

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?

Understanding Graph Neural Networks

Thanks to:
Jorge, Pablo, Egor, Mikael, Juan-Pablo, Floris, Maksimilian, Vicente, Etienne, Domagoj.