A Query Language Perspective on Graph Learning

Floris Geerts (University of Antwerp)

- Database (DB) theoreticians love graphs.

- Database (DB) theoreticians love graphs.
- But so do machine learners (ML).

- Database (DB) theoreticians love graphs.

- But so do machine learners (ML).

How to understand what ML folks are doing with graphs from a DB perspective?

- Graph learning methods can be expressed in specialized graph embedding languages.

.

- Graph learning methods can be expressed in specialized graph embedding languages.

- These languages can be analyzed with regards to expressive power using familiar DB techniques.

Conclusion

- Graph learning methods can be expressed in specialized graph embedding languages.
- These languages can be analyzed with regards to expressive power using familiar DB techniques.

- This results in a better understanding of graph learning methods; and
- forms a bridge between graph learning and DB theory.

A different scope

Previous keynotes at PODS on graph learning:

- word2vec, node2vec, graph2vec, X2vec: Towards a Theory of Vector Embeddings of Structured Data, by Martin Grohe (PODS 2020)
- Databases as Graphs: Predictive Queries for Declarative Machine Learning by Jure Leskovec.

I nevertheless hope to convey some alternative point of view.

1. Preliminaries

Graphs, embeddings and graph learning

Graphs: One definition to rule them all

- A graph $G = (V_G, E_G, L_G)$ with vertex set V_G, edge set $E_G \subseteq V_G \times V_G$, and vertex labelling $L_G : V_G \to \Sigma$ for some set Σ of labels.
- We often assume $\Sigma = \mathbb{R}^d$ for some dimension $d \in \mathbb{N}$.

  Finite set of labels $\mapsto$ hot-one encoding, e.g., labels a, b and c: $(\underbrace{1}_{a}, \underbrace{0}_{b}, \underbrace{0}_{c})$.

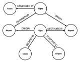

Event Graphs


Image credit: SalientNetworks
Computer Networks

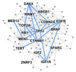

Disease Pathways


Image credit: Medium
Social Networks

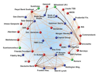


Image credit: Science
Economic Networks


Image credit: Lumen Learning
Communication Networks

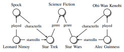


Image credit: Maximilian Nickel et al
Knowledge Graphs

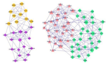


Image credit: ese.wustl.edu
Regulatory Networks

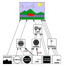


Image credit: math.hws.edu
Scene Graphs

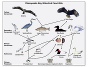


Image credit: Wikipedia
Food Webs


Image credit: Pinterest
Particle Networks


Image credit: visitlondon.com
Underground Networks


Citation Networks


Image credit: Missoula Current News
Internet

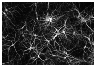


Image credit: The Conversation
Networks of Neurons

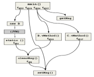


Image credit: ResearchGate
Code Graphs

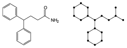


Image credit: MDPI
Molecules

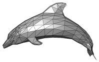


Image credit: Wikipedia
3D Shapes

Source Images: Machine Learning with Graphs course from Jure Leskovec `cs224w.stanford.edu`

Graph embeddings

In this tutorial, graph learning is about learning (partially) unknown graph embeddings.

- Let $\mathcal{G}$ be the class of all graphs.
- Let $\mathbb{Y}$ be an output space.
- A graph embedding is a function of the form

$$\xi : \mathcal{G} \to \mathbb{Y}.$$

Source Images: Stokes et. al, "A Deep Learning Approach to Antibiotic Discovery", Cell 180(4), 2020

Graph embeddings

In this tutorial, graph learning is about learning (partially) unknown graph embeddings.

- Let $\mathcal{G}$ be the class of all graphs.
- Let $\mathbb{Y}$ be an output space.
- A graph embedding is a function of the form

$$\xi : \mathcal{G} \to \mathbb{Y}.$$

For example: prediction of chemical/medical property of molecules

$\xi :$  $\longrightarrow \{\text{yes}, \text{no}\}$

Source Images: Stokes et. al, "A Deep Learning Approach to Antibiotic Discovery", Cell 180(4), 2020

Also, graph learning is about learning unknown vertex embeddings.

- Let $\mathcal{G}$ be the class of all graphs.
- Let $\mathcal{V}$ be the class of all vertices.
- Let $\mathbb{Y}$ be an output space.
- A vertex embedding is a function of the form

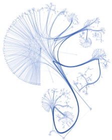
$$\xi : \mathcal{G} \to (\mathcal{V} \to \mathbb{Y}).$$

Also, graph learning is about learning unknown vertex embeddings.

- Let $\mathcal{G}$ be the class of all graphs.
- Let $\mathcal{V}$ be the class of all vertices.
- Let $\mathbb{Y}$ be an output space.
- A vertex embedding is a function of the form

$$\xi : \mathcal{G} \to (\mathcal{V} \to \mathbb{Y}).$$

For example: prediction of the subject of papers in citation network

$\xi :$ 

$$\longrightarrow \begin{cases} \mathit{paper}1 & \longrightarrow \quad \text{computer science} \\ \mathit{paper}2 & \longrightarrow \quad \text{biology} \\ \quad\vdots & \qquad\qquad \vdots \end{cases}$$

More generally, graph learning is about learning unknown p-vertex embeddings.

- Let $\mathcal{G}$ be the class of all graphs.
- Let $\mathcal{V}$ be the class of all vertices.
- Let $\mathbb{Y}$ be an output space.
- A p-vertex embedding is a function of the form

$$\xi : \mathcal{G} \to (\mathcal{V}^p \to \mathbb{Y}).$$

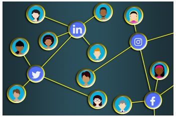
Source Images: http://www.differencebetween.net/

More generally, graph learning is about learning unknown p-vertex embeddings.

- Let $\mathcal{G}$ be the class of all graphs.
- Let $\mathcal{V}$ be the class of all vertices.
- Let $\mathbb{Y}$ be an output space.
- A *p-vertex embedding* is a function of the form

$$\xi : \mathcal{G} \to (\mathcal{V}^p \to \mathbb{Y}).$$

For example: Link prediction in social networks ($p = 2$)

$\xi :$  $\longrightarrow$

$$\begin{cases} (person_1, person_2) & \longrightarrow 1 \text{ (will connect)} \\ (person_3, person_5) & \longrightarrow 0 \text{ (will not connect)} \\ \vdots & \qquad \vdots \end{cases}$$

Embeddings will be at the core of this tutorial.

Invariance

An important requirement is that embeddings should be invariant, i.e., independent of the chosen graph representation.

- A p-vertex embedding ξ is called invariant if for any two graphs G and H in $\mathcal{G}$, for any graph isomorphism $\pi : V_G \to V_H$ from G to H and any p-tuple of vertices $\mathbf{v} \in V_G^p$

$$\xi(G, \mathbf{v}) = \xi\big(\pi(G), \pi(\mathbf{v})\big).$$

- Similar to the genericity requirement for query languages.

An important requirement is that embeddings should be invariant, i.e., independent of the chosen graph representation.

- A p-vertex embedding ξ is called invariant if for any two graphs G and H in $\mathcal{G}$, for any graph isomorphism $\pi : V_G \to V_H$ from G to H and any p-tuple of vertices $\mathbf{v} \in V_G^p$
$$\xi(G, \mathbf{v}) = \xi\big(\pi(G), \pi(\mathbf{v})\big).$$
- Similar to the genericity requirement for query languages.

How are embedding methods specified?

- In the ML community, embedding methods are described by their implementations using linear algebra and other computations on real numbers.
- Crucially, these implementation have learnable parameters/weights.
- Typically, embeddings are defined layer-wise (deep architectures).

- Let σ a non-linear activation function $\mathbb{R} \to \mathbb{R}$ (ReLU, sigmoid, sign,...).
- Vertex set of graph identified with $[n] := \{1, 2, \ldots, n\}$ for some $n \in \mathbb{N}$.
- Output space $\mathbb{Y} = \mathbb{R}^d$ for some $d \in \mathbb{N}$.
- Matrix $\mathbf{F}^{(t)}$ in $\mathbb{R}^{n \times d}$ represents vertex feature computed in layer t.
- In particular, $\mathbf{F}^{(t)}_{v\bullet}$ in $\mathbb{R}^{1 \times d}$ denotes embedding of vertex v.
- "learnable" weight matrices $\mathbf{W}^{(t)}_1, \mathbf{W}^{(t)}_2 \in \mathbb{R}^{d \times d}$ and bias $\mathbf{b}^{(t)} \in \mathbb{R}^{1 \times d}$.

$$\mathbf{F}^{(0)}_{v\bullet} := L_G(v) \qquad \mathbf{F}^{(t)}_{v\bullet} := \sigma\left(\mathbf{F}^{(t-1)}_{v\bullet}\mathbf{W}^{(t)}_1 + \sum_{u \in N_G(v)} \mathbf{F}^{(t-1)}_{u\bullet}\mathbf{W}^{(t)}_2 + \mathbf{b}^{(t)}\right)$$

- By varying weights and biases, an infinite family of vertex embeddings is obtained.

- Let σ a non-linear activation function $\mathbb{R} \to \mathbb{R}$ (ReLU, sigmoid, sign,...).
- Vertex set of graph identified with $[n] := \{1, 2, \ldots, n\}$ for some $n \in \mathbb{N}$.
- Output space $\mathbb{Y} = \mathbb{R}^d$ for some $d \in \mathbb{N}$.
- Matrix $\mathbf{F}^{(t)}$ in $\mathbb{R}^{n \times d}$ represents vertex feature computed in layer t.
- In particular, $\mathbf{F}^{(t)}_{v\bullet}$ in $\mathbb{R}^{1 \times d}$ denotes embedding of vertex v.
- "learnable" weight matrices $\mathbf{W}^{(t)}_1, \mathbf{W}^{(t)}_2 \in \mathbb{R}^{d \times d}$ and bias $\mathbf{b}^{(t)} \in \mathbb{R}^{1 \times d}$.

$$\mathbf{F}^{(0)}_{v\bullet} := L_G(v) \qquad \mathbf{F}^{(t)}_{v\bullet} := \sigma\left(\mathbf{F}^{(t-1)}_{v\bullet}\mathbf{W}^{(t)}_1 + \sum_{u \in N_G(v)} \mathbf{F}^{(t-1)}_{u\bullet}\mathbf{W}^{(t)}_2 + \mathbf{b}^{(t)}\right)$$

- By varying weights and biases, an infinite family of vertex embeddings is obtained.

- Let σ a non-linear activation function $\mathbb{R} \to \mathbb{R}$ (ReLU, sigmoid, sign,...).
- Vertex set of graph identified with $[n] := \{1, 2, \ldots, n\}$ for some $n \in \mathbb{N}$.
- Output space $\mathbb{Y} = \mathbb{R}^d$ for some $d \in \mathbb{N}$.
- Matrix $\mathbf{F}^{(t)}$ in $\mathbb{R}^{n \times d}$ represents vertex feature computed in layer t.
- In particular, $\mathbf{F}^{(t)}_{v\bullet}$ in $\mathbb{R}^{1 \times d}$ denotes embedding of vertex v.
- "learnable" weight matrices $\mathbf{W}_1^{(t)}, \mathbf{W}_2^{(t)} \in \mathbb{R}^{d \times d}$ and bias $\mathbf{b}^{(t)} \in \mathbb{R}^{1 \times d}$.

$$\mathbf{F}^{(0)}_{v\bullet} := L_G(v) \qquad \mathbf{F}^{(t)}_{v\bullet} := \sigma\left(\mathbf{F}^{(t-1)}_{v\bullet}\mathbf{W}_1^{(t)} + \sum_{u \in N_G(v)} \mathbf{F}^{(t-1)}_{u\bullet}\mathbf{W}_2^{(t)} + \mathbf{b}^{(t)}\right)$$

- By varying weights and biases, an infinite family of vertex embeddings is obtained.

- Let σ a non-linear activation function $\mathbb{R} \to \mathbb{R}$ (ReLU, sigmoid, sign,...).
- Vertex set of graph identified with $[n] := \{1, 2, \ldots, n\}$ for some $n \in \mathbb{N}$.
- Output space $\mathbb{Y} = \mathbb{R}^d$ for some $d \in \mathbb{N}$.
- Matrix $\mathbf{F}^{(t)}$ in $\mathbb{R}^{n \times d}$ represents vertex feature computed in layer t.
- In particular, $\mathbf{F}^{(t)}_{v\bullet}$ in $\mathbb{R}^{1 \times d}$ denotes embedding of vertex v.
- "learnable" weight matrices $\mathbf{W}^{(t)}_1, \mathbf{W}^{(t)}_2 \in \mathbb{R}^{d \times d}$ and bias $\mathbf{b}^{(t)} \in \mathbb{R}^{1 \times d}$.

$$\mathbf{F}^{(0)}_{v\bullet} := L_G(v) \qquad \mathbf{F}^{(t)}_{v\bullet} := \sigma\left(\mathbf{F}^{(t-1)}_{v\bullet}\mathbf{W}^{(t)}_1 + \sum_{u \in N_G(v)} \mathbf{F}^{(t-1)}_{u\bullet}\mathbf{W}^{(t)}_2 + \mathbf{b}^{(t)}\right)$$

- By varying weights and biases, an infinite family of vertex embeddings is obtained.

- We can also define a graph embedding
- "learnable" weight matrix $\mathbf{W} \in \mathbb{R}^{d \times d}$ and bias $\mathbf{b} \in \mathbb{R}^{1 \times d}$.
- L is number of layers.

$$\mathbf{F} := \sigma\left(\sum_{v \in V_G} \mathbf{F}^{(L)}_{v\bullet}\mathbf{W} + \mathbf{b}\right)$$

- Easy to see that these GNNs define invariant embeddings.

- But what does "learning an unknown embedding" mean?
- We briefly discuss this in the semi-supervised setting.

Ingredient #1: Training set

We want to learn $\Psi : \mathcal{G} \to (\mathcal{V}^p \to \mathbb{Y})$ but we may only partially know this embedding ...

- Partial knowledge of Ψ is revealed through a training set

$$\mathcal{T} = \Big\{ \big(G_1, \mathbf{v}_1, \Psi(G_1, \mathbf{v}_1) \big), \ldots, \big(G_\ell, \mathbf{v}_\ell, \Psi(G_\ell, \mathbf{v}_\ell) \big) \Big\} \subseteq \mathcal{G} \times \mathcal{V}^p \times \mathbb{Y},$$

with graphs $G_i \in \mathcal{G}$ and p-vertex tuples $\mathbf{v}_i$ in G_i.

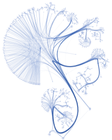
We want to learn $\Psi : \mathcal{G} \to (\mathcal{V}^p \to \mathbb{Y})$ but we may only partially know this embedding ...

- Partial knowledge of Ψ is revealed through a training set

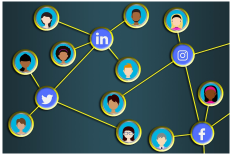
$$\mathcal{T} = \Big\{\big(G_1, \mathbf{v}_1, \Psi(G_1, \mathbf{v}_1)\big), \ldots, \big(G_\ell, \mathbf{v}_\ell, \Psi(G_\ell, \mathbf{v}_\ell)\big)\Big\} \subseteq \mathcal{G} \times \mathcal{V}^p \times \mathbb{Y},$$

with graphs $G_i \in \mathcal{G}$ and p-vertex tuples $\mathbf{v}_i$ in G_i.

(molecule,yes/no)



(cora, paper, topic)



(social,p_x,p_y, yes/no)

Ingredient #2: Hypothesis class

The partially known embedding Ψ will be learned by selecting a good candidate from a class of embeddings.

- An hypothesis class is a collection $\mathcal{F}$ of invariant p-vertex embeddings:

$$\mathcal{F} \subseteq \{\text{all invariant } p\text{-vertex embeddings}\}.$$

Ingredient #2: Hypothesis class

The partially known embedding Ψ will be learned by selecting a good candidate from a class of embeddings.

- An hypothesis class is a collection $\mathcal{F}$ of invariant p-vertex embeddings:

$$\mathcal{F} \subseteq \{\text{all invariant } p\text{-vertex embeddings}\}.$$

For example, $\mathcal{F}$ can be the collection of

- GNN 101's
- Graph kernel methods
- Message-Passing Neural Networks
- Invariant Graph Networks
- Subgraph Networks
- ....

Ingredient #3: Loss function

How to compare Ψ (embedding to be learned) with embeddings ξ from $\mathcal{F}$?

- This is done using a loss function $\mathbf{L} : \mathbb{Y}^2 \to \mathbb{R}$.
- Given graph G, p-vertex tuple $\mathbf{v}$ in our training set $\mathcal{T}$ and embedding ξ in our hypothesis class $\mathcal{F}$,

$$\mathbf{L}\Big(\underbrace{\xi(G,\mathbf{v})}_{\in\mathbb{Y}}, \underbrace{\Psi(G,\mathbf{v})}_{\in\mathbb{Y}}\Big) \in \mathbb{R}$$

measures quality of ξ on the training example $(G,\mathbf{v},\Psi(G,\mathbf{v}))$.

Ingredient #3: Loss function

How to compare Ψ (embedding to be learned) with embeddings ξ from $\mathcal{F}$?

- This is done using a loss function $\mathbf{L} : \mathbb{Y}^2 \to \mathbb{R}$.
- Given graph G, p-vertex tuple $\mathbf{v}$ in our training set $\mathcal{T}$ and embedding ξ in our hypothesis class $\mathcal{F}$,

$$\mathbf{L}\Big(\underbrace{\xi(G,\mathbf{v})}_{\in\mathbb{Y}},\underbrace{\Psi(G,\mathbf{v})}_{\in\mathbb{Y}}\Big) \in \mathbb{R}$$

measures quality of ξ on the training example $(G,\mathbf{v},\Psi(G,\mathbf{v}))$.

Example loss functions: cross entropy, least squares, ...

- Given training data

$$\mathcal{T} = \Big\{\big(G_1, \mathbf{v}_1, \Psi(G_1, \mathbf{v}_1)\big), \ldots, \big(G_\ell, \mathbf{v}_\ell, \Psi(G_\ell, \mathbf{v}_\ell)\big)\Big\} \subseteq \mathcal{G} \times \mathcal{V}^p \times \mathbb{Y},$$

- hypothesis class $\mathcal{F}$, and
- loss function $\mathbf{L}$, return

$$\hat{\xi} := \arg\min_{\xi \in \mathcal{F}} \frac{1}{|\mathcal{T}|} \sum_{(G_i, \mathbf{v}_i, \Psi(G, \mathbf{v}_i)) \in \mathcal{T}} \mathbf{L}\big(\xi(G_i, \mathbf{v}_i), \Psi(G_i, \mathbf{v}_i)\big).$$

- In other words, find a graph, vertex or p-vertex embedding in $\mathcal{F}$ which minimizes the risk (measured by the loss function) on the training data.

Graph learning: Empirical risk minimization

$$\hat{\xi} := \arg\min_{\xi \in \mathcal{F}} \frac{1}{|\mathcal{T}|} \sum_{(G_i, \mathbf{v}_i, \Psi(G, \mathbf{v}_i)) \in \mathcal{T}} \mathbf{L}\big(\xi(G_i, \mathbf{v}_i), \Psi(G_i, \mathbf{v}_i)\big).$$

- Graph learning systems: optimization techniques for finding best hypothesis.
- Typically based on back propagation and gradient descent like methods.

$$\hat{\xi} := \arg\min_{\xi \in \mathcal{F}} \frac{1}{|\mathcal{T}|} \sum_{(G_i, \mathbf{v}_i, \Psi(G, \mathbf{v}_i)) \in \mathcal{T}} \mathbf{L}\big(\xi(G_i, \mathbf{v}_i), \Psi(G_i, \mathbf{v}_i)\big).$$

- Graph learning systems: optimization techniques for finding best hypothesis.
- Typically based on back propagation and gradient descent like methods.

- We will be focussing on:

Expressivity of classes $\mathcal{F}$ of embeddings.

2. Expressive Power

What graph information can be extracted by embedding methods?

Recall our GNN 101's.

- Which graph or vertex embeddings can they express?
- Which graph or vertex embeddings can they approximate?
- Which graphs or vertices can be discriminated/distinguished?

Answers to these questions may reveal

- what graph information is used by embedding methods;
- which embeddings could – in principle – be learned; and
- whether more powerful embedding methods may be needed for the application at hand.

Let $\Psi : \mathcal{G} \to (\mathcal{V}^p \to \mathbb{Y})$ be a p-vertex embedding and let $\mathcal{F}$ be a class of embeddings and let $\mathcal{C}$ be a subset of $\mathcal{G}$.

- $\mathcal{F}$ can $\mathcal{C}$-express Ψ if

$$\exists \xi \in \mathcal{F}, \forall G \in \mathcal{C}, \forall \mathbf{v} \in V_G^p : \Psi(G, \mathbf{v}) = \xi(G, \mathbf{v}).$$

- $\mathcal{F}$ can $\mathcal{C}$-approximate Ψ if

$$\forall \epsilon > 0, \exists \xi_\epsilon \in \mathcal{F}, \forall G \in \mathcal{C}, \forall \mathbf{v} \in V_G^p : \|\Psi(G, \mathbf{v}) - \xi_\epsilon(G, \mathbf{v})\| < \epsilon.$$

  for some norm $\|\cdot\|$ on $\mathbb{Y}$.

If $\mathcal{C} = \mathcal{G}$ we just say express or approximate.

Separation power measures how well $\mathcal{F}$ can separate different inputs.

- As before, $\mathcal{F}$ be a class of p-vertex embeddings.
- The separation power of $\mathcal{F}$ is captured by equivalence relation $\rho(\mathcal{F})$ on $\mathcal{G} \times \mathcal{V}^p$:

$$\big(G, \mathbf{v}; H, \mathbf{w}\big) \in \rho(\mathcal{F}) \iff \forall \xi \in \mathcal{F} : \xi(G, \mathbf{v}) = \xi(H, \mathbf{w}).$$

- In other words, $(G, \mathbf{v})$ and $(H, \mathbf{w})$ are in $\rho(\mathcal{F})$ when these cannot be separated by any embedding in $\mathcal{F}$
- Similar to the notion of indistinguishability for logics and query languages.

Separation power

Strongest power

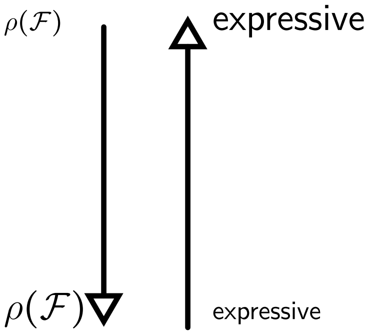
- $\mathcal{F}$ powerful enough to distinguish non-isomorphic graphs:

$$\rho(\mathcal{F}) = \{\text{all pairs of isomorphic graphs}\}.$$

Weakest power

- $\mathcal{F}$ consisting of constant functions.

$$\rho(\mathcal{F}) = \{\text{all pairs of graphs}\}.$$



Separation power allows for comparing totally different embedding methods by means of subset relationship of their separation power.

- Primary focus has been on separation power.
- Aim is to provide a characterization of when $(G, H) \in \rho(\mathcal{F})$ holds.
- For example,

Theorem

$$\rho(\text{GNNs 101}) = \rho(\text{color refinement}).$$

Shown in the - by now - seminal paper in the area of graph learning[1].

[1] ☞ Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. AAAI (2019)

- Characterizing $\rho(\mathcal{F})$ is a bit of hot potato game.
- Try to find characterizations of $\rho(\mathcal{F})$ that are insightful.
- For example, color refinement has been well-studied, many properties thereof are known.
- In particular, G and H are in ρ(color refinement) if and only if

$$\hom(T, G) = \hom(T, H)$$

for all trees T. Here, $\hom(T, G)$ counts homomorphisms from T to G.[2]
- GNNs 101 can only leverage tree-based information present in the graphs.

[2] ☞Dell, Grohe and Rattan: Lovász Meets Weisfeiler and Leman. ICALP 2018

- Characterizing $\rho(\mathcal{F})$ is a bit of hot potato game.
- Try to find characterizations of $\rho(\mathcal{F})$ that are insightful.
- For example, color refinement has been well-studied, many properties thereof are known.
- In particular, G and H are in ρ(color refinement) if and only if

$$\text{hom}(T, G) = \text{hom}(T, H)$$

  for all trees T. Here, $\text{hom}(T, G)$ counts homomorphisms from T to G.[2]
- GNNs 101 can only leverage tree-based information present in the graphs.

[2] ☞ Dell, Grohe and Rattan: Lovász Meets Weisfeiler and Leman. ICALP 2018

Two additional reasons for studying separation power:

1. Close connection between separation power $\rho(\mathcal{F})$ and ability for $\mathcal{F}$ to approximate functions.

2. Close connection between $\rho(\mathcal{F})$ and Vapnik-Chervonenkis (VC) dimension of $\mathcal{F}$. This implies properties of generalization aspects of $\mathcal{F}$.[3]

[3] ☞WL meet VC. Morris, G*, Tönshoff and Grohe. ICML 2023

- $\mathcal{C}$ be a compact subset of $\mathcal{G}$.
- Assume embeddings in $\mathcal{F}$ are continuous.
- We can use tools from analysis and topology.

Theorem (General version of Stone-Weierstrass)

If $\mathcal{F}$ is closed under linear combinations and product, then $\mathcal{F}$ can $\mathcal{C}-$approximate any continuous embedding $\Psi : \mathcal{G} \to \mathbb{R}$ such that

$$\rho(\mathcal{F}) \subseteq \rho(\{\Psi\})$$

holds.

- Can be generalized to vertex embeddings and output spaces $\mathbb{Y} = \mathbb{R}^d$. [4,5]

[4] ☞ Azizian and Lelarge. Characterizing the Expressive Power of Invariant and Equivariant GNNs, ICLR 2021

[5] ☞ G* and Reutter. Expressiveness and Approximation Properties of GNNs. ICLR 2022

Theorem

On compact sets of graphs, GNNs 101 can approximate any continuous embedding Ψ whose separation power is bounded by color refinement.

Follows from ρ(GNNs 101) = ρ(color refinement), universality theorem of neural networks (to approximate product), and Stone-Weierstrass. (Alternative proof based on homomorphism counts.[6])

[6] ☞ Nguyen and Maehara. Graph Homomorphism Convolution. ICML 2020.

Consequence of Stone-Weierstrass:[7]

Theorem

For a class $\mathcal{F}$ to be able to approximate any invariant embedding on a compact set of graphs, $\mathcal{F}$ needs to be able to separate any two non-isomorphic graphs.

[7] ☞ Chen, Villar, Chen and Bruna. 2019. On the Equivalence Between Graph Isomorphism Testing and Function Approximation With GNNs. Neurips 2019.

Promised query language perspective is coming up in a few moments.

Expressive power in ML community

- Every week new embedding methods are being proposed.
- Continuous stream of papers on arxiv.
- Has become standard to analyze separation power of new methods.
- This is done often in an ad hoc way.

This is where the language approach comes in the picture.

A small selection of methods...

GraphSage GINs GCNs SGNs

GATs GatedGCNs extended GINs 2-IGNs ChebNet ...

Walk GNNs 2WL-GNNs ring-GNNs 1-Dropout GNNs

Id-aware GNNs CayleyNet 3-IGNs 2-FGNNs ...

kWL-GNNs k-FGNNs (k+1)-IGNs GSNs k-Dropout GNNs ...

1. View embedding methods as queries in some graph embedding language
2. Transfer our understanding of separation power of these languages back to embedding methods.

Recipe

- A new embedding method just needs to be cast in the embedding language to know a bound on its expressive power.

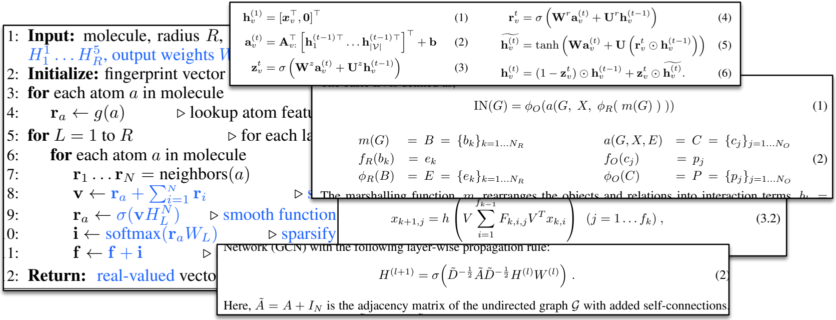
3. Embedding Language #1: Message Passing Neural Networks

Message Passing Neural Networks (MPNNs)

- We go back in time (around 2016) when embedding methods like
  - Graph convolutional networks (Duvenaud et al. 2016, Kearnes et al. 2016),
  - Gated GNNs (Li et al. 2016),
  - Interaction Networks (Battaglia et al. 2016),
  - Deep tensor neural networks (Schütt et al. 2017), and
  - Laplacian based graph convolutional networks (Bruna et al. 2013, Defferrard et al. 2016, Kipf & Welling 2016)

  were "hot".

- In 2017, Gilmer et al.[8] looked at the specifications in those papers



and proposed a first unifying framework for specifying embedding methods: Message Passing Neural Networks

[8] ☞ Gilmer, Schoenholz, Riley, Vinyals, Dahl. Neural Message Passing for Quantum Chemistry, Neurips, 1263–1272 (2017)

Message Passing Neural Networks (MPNNs)

- Liberally interpreted, Gilmer et al. proposed an inductive way of defining vertex and graph embeddings.
- Indeed, one has initial vertex embeddings

$$\xi : \mathcal{G} \to (\mathcal{V} \to \mathbb{R}^d) : (G, v) \mapsto \nu_G(v)$$

for some encoding of the vertex label in $\nu_G(v) \in \Sigma$ in $\mathbb{R}^d$.

Message Passing Neural Networks (MPNNs)

- Liberally interpreted, Gilmer et al. proposed an inductive way of defining vertex and graph embeddings.
- Indeed, one has initial vertex embeddings

$$\xi : \mathcal{G} \to (\mathcal{V} \to \mathbb{R}^d) : (G, v) \mapsto \nu_G(v)$$

for some encoding of the vertex label in $\nu_G(v) \in \Sigma$ in $\mathbb{R}^d$.

- Then, let $\xi' : \mathcal{G} \to (\mathcal{V} \to \mathbb{R}^{d'})$ be an old vertex embedding. A new vertex embedding $\xi : \mathcal{G} \to (\mathcal{V} \to \mathbb{R}^d)$ can be obtained by:

- Liberally interpreted, Gilmer et al. proposed an inductive way of defining vertex and graph embeddings.
- Indeed, one has initial vertex embeddings

$$\xi : \mathcal{G} \to (\mathcal{V} \to \mathbb{R}^d) : (G, v) \mapsto \nu_G(v)$$

for some encoding of the vertex label in $\nu_G(v) \in \Sigma$ in $\mathbb{R}^d$.

- Then, let $\xi' : \mathcal{G} \to (\mathcal{V} \to \mathbb{R}^{d'})$ be an old vertex embedding. A new vertex embedding $\xi : \mathcal{G} \to (\mathcal{V} \to \mathbb{R}^d)$ can be obtained by:

$$(G, v) \mapsto \xi(G, v) := \sum_{u \in N_G(v)} \xi'(G, u)$$

- Liberally interpreted, Gilmer et al. proposed an inductive way of defining vertex and graph embeddings.
- Indeed, one has initial vertex embeddings

$$\xi : \mathcal{G} \to (\mathcal{V} \to \mathbb{R}^d) : (G, v) \mapsto \nu_G(v)$$

for some encoding of the vertex label in $\nu_G(v) \in \Sigma$ in $\mathbb{R}^d$.

- Then, let $\xi' : \mathcal{G} \to (\mathcal{V} \to \mathbb{R}^{d'})$ be an old vertex embedding. A new vertex embedding $\xi : \mathcal{G} \to (\mathcal{V} \to \mathbb{R}^d)$ can be obtained by:

$$(G, v) \mapsto \xi(G, v) := \underbrace{\textbf{Update}}_{\substack{\text{any function}\\ \mathbb{R}^{2d'} \to \mathbb{R}^d}} \left(\xi'(G, v), \sum_{u \in N_G(v)} \xi'(G, u)\right)$$

- Finally, one can also construct graph embeddings $\xi : \mathcal{G} \to \mathbb{R}^d$ from a vertex embedding $\xi' : \mathcal{G} \to (\mathcal{V} \to \mathbb{R}^{d'})$ as follows:

$$G \mapsto \xi(G) := \sum_{v \in V_G} \xi'(G, v)$$

- Finally, one can also construct graph embeddings $\xi : \mathcal{G} \to \mathbb{R}^d$ from a vertex embedding $\xi' : \mathcal{G} \to (\mathcal{V} \to \mathbb{R}^{d'})$ as follows:

$$G \mapsto \xi(G) := \underbrace{\textbf{Readout}}_{\substack{\text{any function} \\ \mathbb{R}^{d'} \to \mathbb{R}^d}}\Big(\sum_{v \in V_G} \xi'(G, v)\Big)$$

Message Passing Neural Networks (MPNNs)

- Finally, one can also construct graph embeddings $\xi : \mathcal{G} \to \mathbb{R}^d$ from a vertex embedding $\xi' : \mathcal{G} \to (\mathcal{V} \to \mathbb{R}^{d'})$ as follows:

$$G \mapsto \xi(G) := \underbrace{\textbf{Readout}}_{\substack{\text{any function} \\ \mathbb{R}^{d'} \to \mathbb{R}^d}}\Big(\sum_{v \in V_G} \xi'(G, v)\Big)$$

Easy exercise: The GNNs 101 we have seen before are MPNNs.

Layer-based vector embedding computation

$$\mathbf{F}^{(0)} := \text{Initial vertex labels} \quad \mathbf{F}^{(t)} := \sigma\left(\mathbf{F}^{(t-1)}\mathbf{W}_1^{(t)} + \mathbf{A}\mathbf{F}^{(t-1)}\mathbf{W}_2^{(t)} + \mathbf{B}^{(t)}\right)$$

- σ is a non-linear activation function $\mathbb{R} \to \mathbb{R}$;
- $\mathbf{A}$ is adjacency matrix in $\mathbb{R}^{n \times n}$ of a graph;
- $\mathbf{F}^{(t)}$ vertex embedding in $\mathbb{R}^{n \times d}$ computed in layer t; and
- learnable weight matrices $\mathbf{W}_1^{(t)}, \mathbf{W}_2^{(t)} \in \mathbb{R}^{d \times d}$ and bias $\mathbf{B}^{(t)} \in \mathbb{R}^{n \times d}$.

$$\mathbf{F}_{v\bullet}^{(t)} := \sigma\left(\mathbf{F}_{v\bullet}^{(t-1)}\mathbf{W}_1^{(t)} + \sum_{u \in N_G(v)} \mathbf{F}_{u\bullet}^{(t-1)}\mathbf{W}_2^{(t)} + \mathbf{b}^{(t)}\right) \quad \mathbf{F} := \sigma\left(\sum_{v \in V_G} \mathbf{F}^{(L)}\mathbf{W} + \mathbf{b}\right)$$

Let's put on our database glasses

Let's put on our database glasses (you can find those under your seats).

We can turn MPNNs into a specification language for vertex and graph embeddings.

- We fix the output space $\mathbb{Y}$ to be $\bigcup_d \mathbb{R}^d$.
- We take two variables x_1 and x_2.
- Inductively define MPNN expressions φ.

We can turn MPNNs into a specification language for vertex and graph embeddings.

- We fix the output space $\mathbb{Y}$ to be $\bigcup_d \mathbb{R}^d$.
- We take two variables x_1 and x_2.
- Inductively define MPNN expressions φ. Each expression φ comes with:
  - a dimension $dim(\varphi) \in \mathbb{N}$ and
  - a set of free variables $\mathrm{fv}(\varphi)$.

We can turn MPNNs into a specification language for vertex and graph embeddings.

- We fix the output space $\mathbb{Y}$ to be $\bigcup_d \mathbb{R}^d$.
- We take two variables x_1 and x_2.
- Inductively define MPNN expressions φ. Each expression φ comes with:
  - a dimension $\mathit{dim}(\varphi) \in \mathbb{N}$ and
  - a set of free variables $\mathrm{fv}(\varphi)$.
- Semantics: p-vertex embeddings $\xi_\varphi : \mathcal{G} \to (\mathcal{V}^p \to \mathbb{Y})$ (p determined by number free variables, output space $\mathbb{R}^d$ by dimension of φ).

- Initially, atomic MPNN expressions are of the form

$$\varphi(x_i) := \mathrm{lab}_j(x_i)$$

with free variable x_i, $j = 1, 2, 3, \ldots$ and of dimension 1.

- Initially, atomic MPNN expressions are of the form

$$\varphi(x_i) := \mathsf{lab}_j(x_i)$$

with free variable x_i, $j = 1, 2, 3, \ldots$ and of dimension 1.

For example, let

$$\varphi(x_1) := \mathsf{Lab}_2(x_1)$$

and G a graph with vertex labelling $\nu_G : V_G \to \mathbb{R}^3$. The corresponding semantics is the vertex embedding

$$\xi_\varphi : (G, v) \mapsto (\nu_G(v))_2 \in \mathbb{R}$$

i.e., the second component of $\nu_G(v) \in \mathbb{R}^3$.

We close under application of functions coming from some set Ω.

- Consider MPNN expressions $\varphi_1(x_i), \ldots, \varphi_\ell(x_i)$ with free variable x_i and of dimension $d_1, \ldots, d_\ell$, respectively.
- Take a function $\mathbf{F} : \mathbb{R}^{d_1+\cdots+d_\ell} \to \mathbb{R}^d$ in Ω. Then,

$$\varphi(x_i) := \mathbf{F}\big(\varphi_1(x_i), \ldots, \varphi_\ell(x_i)\big)$$

is again an MPNN expression, of dimension d and free variable x_i.

We close under application of functions coming from some set Ω.

- Consider MPNN expressions $\varphi_1(x_i), \ldots, \varphi_\ell(x_i)$ with free variable x_i and of dimension $d_1, \ldots, d_\ell$, respectively.
- Take a function $\mathbf{F} : \mathbb{R}^{d_1+\cdots+d_\ell} \to \mathbb{R}^d$ in Ω. Then,

$$\varphi(x_i) := \mathbf{F}\big(\varphi_1(x_i), \ldots, \varphi_\ell(x_i)\big)$$

  is again an MPNN expression, of dimension d and free variable x_i.

Example:

$$\varphi(x_1) := \mathsf{ReLU}\big(\varphi'(x_1)\big) \text{ with } \mathsf{ReLU}(x) := \max\{0, x\} \text{ in } \Omega$$

with semantics

$$\xi_\varphi : (G, v) \mapsto \mathsf{ReLU}\big(\xi_{\varphi'}(G, v)\big)$$

We close under restricted application of aggregation functions from a set Θ.

- Let $\varphi_1(x_1)$ and $\varphi_2(x_2)$ expressions with free variables x_1 and x_2 of dimension d_1 and d_2, respectively.
- Let $\boldsymbol{\theta}$ be any aggregate function from bags of elements in $\mathbb{R}^{d_1+d_2}$ to $\mathbb{R}^d$.
- Then,

$$\varphi(x_1) := \mathrm{agg}_{x_2}^{\boldsymbol{\theta}}\Big(\varphi_1(x_1), \varphi_2(x_2) \;\Big|\; \underbrace{E(x_1, x_2)}_{\text{edge relation}}\Big)$$

is an MPNN expression of dimension d and free variable x_1 (similarly with roles of x_1 and x_2 reversed.)

We close under restricted application of aggregation functions from a set Θ.

- Let $\varphi_1(x_1)$ and $\varphi_2(x_2)$ expressions with free variables x_1 and x_2 of dimension d_1 and d_2, respectively.
- Let $\boldsymbol{\theta}$ be any aggregate function from bags of elements in $\mathbb{R}^{d_1+d_2}$ to $\mathbb{R}^d$.
- Then,

$$\varphi(x_1) := \mathrm{agg}_{x_2}^{\boldsymbol{\theta}}\Big(\varphi_1(x_1), \varphi_2(x_2) \;\Big|\; \underbrace{E(x_1, x_2)}_{\text{edge relation}}\Big)$$

is an MPNN expression of dimension d and free variable x_1 (similarly with roles of x_1 and x_2 reversed.)

Example:

$$\varphi(x_1) := \mathrm{agg}_{x_2}^{\mathrm{sum}}\big(\varphi_1(x_1), \varphi_2(x_2) | E(x_1, x_2)\big)$$

and corresponding embedding

$$\xi_\varphi : (G, v) \mapsto \sum_{(v,u)\in E_G} (\xi_{\varphi_1}(G, v), \xi_{\varphi_2}(G, u))$$

We can also express graph embeddings.

- Let $\varphi'(x_1)$ be an MPNN expression with free variable x_1 and of dimension d'.
- Let $\boldsymbol{\theta}$ be an aggregation function from bags of elements in $\mathbb{R}^{d'}$ to $\mathbb{R}^{d}$.
- Then,

$$\varphi := \mathsf{agg}_{x_1}^{\boldsymbol{\theta}}\Big(\varphi'(x_1)\Big)$$

is an MPNN expression of dimension d and no free variables.

We can also express graph embeddings.

- Let $\varphi'(x_1)$ be an MPNN expression with free variable x_1 and of dimension d'.
- Let $\boldsymbol{\theta}$ be an aggregation function from bags of elements in $\mathbb{R}^{d'}$ to $\mathbb{R}^{d}$.
- Then,

$$\varphi := \mathrm{agg}_{x_1}^{\boldsymbol{\theta}}\Big(\varphi'(x_1)\Big)$$

is an MPNN expression of dimension d and no free variables.

Example:

$$\varphi := \mathrm{agg}_{x_1}^{\mathrm{sum}}\big(\varphi_1(x_1)\big)$$

and corresponding embedding

$$\xi_\varphi : G \mapsto \sum_{v \in V_G} (\xi_{\varphi_1}(G, v)$$

- We have thus defined a language $\mathsf{MPNN}(\Omega, \Theta)$.
- A very limited fragment of calculus with aggregates.[9]
- It differs from classical MPNNs because in those one restricts how function application and aggregation interleave:

$$\varphi^{(t)}(x_1) := \mathbf{F}^{(t)}\Big(\varphi^{(t-1)}(x_1), \mathsf{agg}_{x_2}^{\boldsymbol{\theta}^{(t)}}\big(\varphi^{(t-1)}(x_2) \mid E(x_1, x_2)\big)\Big)$$

[9] ☞ Hella, Libkin, Nurmonen and Wong, Logics with aggregate operators, JACM 2001.

- Existing architectures can be easily cast as $\mathrm{MPNN}(\Omega, \Theta)$ expressions, due to more flexible definition when compared to classical MPNNs.

GraphSage GINs GCNs SGNs

GATs GatedGCNs extended GINs 2-IGNs ChebNet ...

Walk GNNs 2WL-GNNs ring-GNNs 1-Dropout GNNs

Id-aware GNNs CayleyNet 3-IGNs 2-FGNNs ...

kWL-GNNs k-FGNNs (k+1)-IGNs GSNs k-Dropout GNNs ...

What about separation power of $\mathsf{MPNN}(\Omega, \Theta)$?

Color refinement: Iteratively computes a coloring of vertices of a graph:

1. Initialization: all vertices have their original colors (labels)
2. Refinement Step: two vertices v and w get different colors if the there is a color c such that v and w have a different number of neighbors of color c.

This process terminates and a graph will get a color based on the multiset of colors of all its vertices.

ρ(color refinement) contains pairs of graphs/vertices with the same coloring.

Color refinement: Iteratively computes a coloring of vertices of a graph:

1. Initialization: all vertices have their original colors (labels)
2. Refinement Step: two vertices v and w get different colors if the there is a color c such that v and w have a different number of neighbors of color c.

This process terminates and a graph will get a color based on the multiset of colors of all its vertices.

ρ(color refinement) contains pairs of graphs/vertices with the same coloring.

Theorem

For any Ω and Θ, $\rho_{0/1}\big(\text{color refinement}\big) \subseteq \rho_{0/1}\big(MPNN(\Omega, \Theta)\big)$.

- For MPNNs, this was shown in the seminal papers[10,11] and then expanded.[12]
- This can also be shown using the correspondence

$$\rho(\text{color refinement}) = \rho(\text{guarded } C_2)$$

and elimination of function and aggregation functions by detour to infinitary counterparts of guarded C_2.[13]

[10] ☞ Xu, Hu, Leskovec, Jegelka. How Powerful are Graph Neural Networks? ICLR (2019)

[11] ☞ Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. AAAI (2019)

[12] ☞ G* and Reutter. Expressiveness and Approximation Properties of GNNs. ICLR 2022

[13] ☞ Hella, Libkin, Nurmonen and Wong, Logics with aggregate operators, JACM 2001.

Which functions are needed to match color refinement in separation power?

Theorem

if Ω contains concatenation, linear combinations and non-linear activation functions and Θ consists of summation, then

$$\rho_{0/1}\big(\textit{color refinement}\big) = \rho_{0/1}\big(\mathit{MPNN}(\Omega, \Theta)\big)$$

- Shown by explicit construction of GNNs 101.[14]

[14] ☞ Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. AAAI (2019)

MPNN(Ω, Θ): Approximation power

Theorem

For sufficiently rich sets Ω of functions and compact $\mathcal{C}$ sets of $\mathcal{G}$, MPNN(Ω, sum) can $\mathcal{C}$-approximate any embedding $\Psi : \mathcal{G} \to \mathbb{R}$ satisfying $\rho(\text{color refinement}) \subseteq \rho(\{\Psi\})$.

- For example: Ω is rich enough when it is mlp-closed. That is, for any $q \in \mathbb{N}$, any multilayered perceptron[15] $\mathsf{mlp} : \mathbb{R}^q \to \mathbb{R}$ and any functions $f_1, \ldots, f_q$ already in Ω,

$$\mathsf{mlp}(f_1, \ldots, f_q)$$

is also in $\mathcal{F}$.

[15] MLPs: Layered architectures $\mathbf{F}^{(t)} := \sigma(\mathbf{W}^{(t)}\mathbf{F}^{(t-1)} + \mathbf{b}^{(t)})$, $\mathbf{F}^{(0)} := \mathbf{x}$ with learnable weight matrices $\mathbf{W}^{(t)}$, bias vectors $\mathbf{b}^{(t)}$, and activation functions σ.

MPNN(Ω, Θ): True expressiveness

Can we get rid of compact domain?

Assuming graphs with discrete labels.[16]

Theorem

MPNN(Ω, Θ) can express any unary query expressible in graded modal logic.

GNNs 101 already suffice for this.

Theorem

If a first-order logic unary query is expressible in MPNN(Ω, Θ), then it is a query expressible in graded modal logic.

[16] ☞ Barceló, Kostylev, Monet, Přez, Reutter and Silva. The Logical Expressiveness of Graph Neural Networks. ICLR 2020

MPNN(Ω, Θ): Normal forms

- Can any MPNN(Ω, Θ) be converted into a "normal form" MPNN(Ω, Θ)

$$\varphi^{(t)}(x_1) := \mathbf{F}^{(t)}\Big(\varphi^{(t-1)}(x_1), \mathrm{agg}_{x_2}^{\boldsymbol{\theta}^{(t)}}\big(\varphi^{(t-1)}(x_2) \mid E(x_1, x_2)\big)\Big)$$

- Important for implementation purposes!
- Partial results when Ω consists of linear combinations and activation functions σ, and Θ is summation.[17]

Theorem

- *Every MPNN(Ω, sum) is equivalent to a normal form MPNN if σ = ReLU.*
- *On compact domain $\mathcal{C}$, normal form ReLU MPNNs $\mathcal{C}$-approximate embeddings in MPNN(Ω, sum).*

[17] ☞ G*, Steegmans and Van den Bussche: On the Expressive Power of Message-Passing Neural Networks as Global Feature Map Transformers. FoIKS 2022

- Many other embedding methods exist which are not in $\mathrm{MPNN}(\Omega, \Theta)$.

Only "simple" GNNs are MPNNs :-(

GraphSage GINs GCNs SGNs

GATs GatedGCNs extended GINs 2-IGNs ChebNet ...

Walk GNNs 2WL-GNNs ring-GNNs 1-Dropout GNNs

Id-aware GNNs CayleyNet 3-IGNs 2-FGNNs ...

kWL-GNNs k-FGNNs (k+1)-IGNs GSNs k-Dropout GNNs ...

How to analyze all other embedding methods?

4. Embedding Language #2: $\mathbf{GEL}(\Omega, \Theta)$

and its finite variable fragments

We expand the language of $\mathrm{MPNN}(\Omega, \Theta)$: $\mathrm{GEL}(\Omega, \Theta)$

1. More variables $x_1, x_2, \ldots$;
2. More atomic MPNNs;
3. More general function and aggregation applications.

We expand the language of $\mathsf{MPNN}(\Omega, \Theta)$: $\mathsf{GEL}(\Omega, \Theta)$

1. More variables $x_1, x_2, \ldots$;
2. More atomic MPNNs;
3. More general function and aggregation applications.

Atomic GEL expressions

Label: $\mathrm{Lab}_j(x_i)$ of dimension 1, free variable x_i;

Edge: $E(x_i, x_j)$ of dimension 1, free variables x_i and x_j;

Equality: $\mathbf{1}[x_i \,\mathrm{op}\, x_j]$ with $\mathrm{op} \in \{=, \neq\}$, of dimension 1 and free variables x_i and x_j.

Atomic GEL expressions

Label: $\mathrm{Lab}_j(x_i)$ of dimension 1, free variable x_i;

Edge: $E(x_i, x_j)$ of dimension 1, free variables x_i and x_j;

Equality: $\mathbf{1}[x_i \,\mathrm{op}\, x_j]$ with $\mathrm{op} \in \{=, \neq\}$, of dimension 1 and free variables x_i and x_j.

Example:

$$\varphi(x_1, x_2) := E(x_1, x_2) \text{ and } \psi(x_1, x_2) := \mathbf{1}[x_1 \,\mathrm{op}\, x_2]$$

and corresponding 2-vertex embeddings

$$\xi_\varphi : (G, v, w) \mapsto \begin{cases} 1 & \text{if } (v, w) \in E_G \\ 0 & \text{otherwise.} \end{cases} \text{ and } \xi_\psi : (G, v, w) \mapsto \begin{cases} 1 & \text{if } v \,\mathrm{op}\, w \\ 0 & \text{otherwise.} \end{cases}$$

We close GEL under function applications with function in Ω.

- As before, Ω still set of functions $\mathbb{R}^{d'} \rightarrow \mathbb{R}^{d}$.
- Consider expressions $\varphi_1(\mathbf{x}_1), \ldots, \varphi_\ell(\mathbf{x}_\ell)$ with free variables $\mathbf{x}_i$ and of dimension $d_1, \ldots, d_\ell$, respectively.
- Take a function $\mathbf{F} : \mathbb{R}^{d_1+\cdots+d_\ell} \rightarrow \mathbb{R}^{d}$ in Ω. Then,

$$\varphi(\mathbf{x}) := \mathbf{F}\big(\varphi_1(\mathbf{x}_1), \ldots, \varphi_\ell(\mathbf{x}_\ell)\big)$$

is again an MPNN expression, of dimension d and free variables $\mathbf{x} := \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_\ell$.

We close GEL under function applications with function in Ω.

- As before, Ω still set of functions $\mathbb{R}^{d'} \to \mathbb{R}^d$.
- Consider expressions $\varphi_1(\mathbf{x}_1), \ldots, \varphi_\ell(\mathbf{x}_\ell)$ with free variables $\mathbf{x}_i$ and of dimension $d_1, \ldots, d_\ell$, respectively.
- Take a function $\mathbf{F} : \mathbb{R}^{d_1+\cdots+d_\ell} \to \mathbb{R}^d$ in Ω. Then,

$$\varphi(\mathbf{x}) := \mathbf{F}\big(\varphi_1(\mathbf{x}_1), \ldots, \varphi_\ell(\mathbf{x}_\ell)\big)$$

is again an MPNN expression, of dimension d and free variables $\mathbf{x} := \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_\ell$.

Example:

$$\varphi(x_1, x_2, x_3) := f_\times\Big(E(x_1, x_2), f_\times\big(E(x_1, x_2), E(x_2, x_3)\big)\Big)$$

with $f_\times : \mathbb{R}^2 \to \mathbb{R} : (a, b) \to a \times b$. Then, we obtain a 3-vertex embedding

$$\xi_\varphi : (G, u, v, w) \mapsto \begin{cases} 1 & (u, v), (u, w), \text{ and } (v, w) \in E_G \\ 0 & \text{otherwise.} \end{cases}$$

We close GEL under aggregation with aggregations in Θ.

- Let $\varphi_1(\mathbf{x},\mathbf{y})$ and $\varphi_2(\mathbf{x},\mathbf{y})$ expressions with free variables $(\mathbf{x},\mathbf{y})$ and of dimension d_1 and d_2, respectively.
- Let $\boldsymbol{\theta}$ be any aggregate function from bags of elements in $\mathbb{R}^{d_1}$ to $\mathbb{R}^{d}$.
- Then,

$$\varphi(\mathbf{x}) := \mathsf{agg}_{\mathbf{y}}^{\boldsymbol{\theta}}\Big(\varphi_1(\mathbf{x},\mathbf{y}) \,\Big|\, \varphi_2(\mathbf{x},\mathbf{y})\Big)$$

  is an expression of dimension d and free variables $\mathbf{x}$.

We close GEL under aggregation with aggregations in Θ.

- Let $\varphi_1(\mathbf{x},\mathbf{y})$ and $\varphi_2(\mathbf{x},\mathbf{y})$ expressions with free variables $(\mathbf{x},\mathbf{y})$ and of dimension d_1 and d_2, respectively.
- Let $\boldsymbol{\theta}$ be any aggregate function from bags of elements in $\mathbb{R}^{d_1}$ to $\mathbb{R}^{d}$.
- Then,

$$\varphi(\mathbf{x}) := \mathsf{agg}_{\mathbf{y}}^{\boldsymbol{\theta}}\Big(\varphi_1(\mathbf{x},\mathbf{y}) \;\Big|\; \varphi_2(\mathbf{x},\mathbf{y})\Big)$$

is an expression of dimension d and free variables $\mathbf{x}$.

Semantics

$$\xi_{\varphi} : (G,\mathbf{v}) \mapsto \boldsymbol{\theta}\Big(\Big\{\!\!\Big\{\xi_{\varphi_1}(G,\mathbf{v},\mathbf{w}) \;\Big|\; \mathbf{w} \in V_G^p \text{ s.t. } \xi_{\varphi_2}(G,\mathbf{v},\mathbf{w}) \neq \mathbf{0}\Big\}\!\!\Big\}\Big).$$

with $p := |\mathbf{y}|$.

Important special fragments:

- $\mathsf{GEL}_k(\Omega, \Theta)$: only k variables $x_1, \ldots, x_k$ may be used;
- $\mathsf{GGEL}_2(\Omega, \Theta)$: guarded fragment of GEL_2 in which aggregation and function application are restricted $= \mathsf{MPNN}(\Omega, \Theta)$

Validation

$\mathsf{MPNN}(\Omega,\Theta)$ $\mathsf{GEL}_2(\Omega,\Theta)$ $\mathsf{GEL}_3(\Omega,\Theta)$ $\mathsf{GEL}_k(\Omega,\Theta)$

GraphSage GINs GCNs SGNs

GATs GatedGCNs extended GINs 2-IGNs ChebNet ...

Walk GNNs 2WL-GNNs ring-GNNs 1-Dropout GNNs

Id-aware GNNs CayleyNet 3-IGNs 2-FGNNs ...

kWL-GNNs k-FGNNs (k+1)-IGNs GSNs k-Dropout GNNs ...

For many of these GNNs, their layer definitions translate naturally into expressions in our language.

What can we say about separation power of $\mathrm{GEL}_k(\Omega, \Theta)$?

The k-dimensional Weisfeiler-Leman algorithm:[18,19] Iteratively computes a coloring of k-tuples of vertices of a graph

Intuitively, it can be seen as color refinement on a k-fold product of a graph

Again, has been subject to many theoretical studies and is used in graph isomorphism algorithms

$\rho(k\text{-WL})$ contains pairs of k-tuples of vertices with the same k-WL coloring

It is known:[20] $\rho(\text{CR}) \supseteq \rho(\text{1-WL}) \supsetneq \rho(\text{2-WL}) \supsetneq \rho(\text{3-WL}) \supsetneq \cdots \supsetneq \rho(\text{graph iso})$.

[18] ☞ Grohe. The logic of graph neural networks LICS, 1–17 (2021)

[19] ☞ Morris, Lipman, Maron, Rieck, Kriege, Grohe, Fey, Borgwardt. Weisfeiler and Leman go Machine Learning: The Story So Far. CoRR abs/2112.09992 (2021)

[20] ☞ Cai, Fürer, Immerman: An optimal lower bound on the number of variables for graph identification. Comb. 12(4):389-410 (1992)

Separation power of $\mathsf{GEL}_k(\Omega, \Theta)$

Theorem

For any Ω and Θ, $\rho(k\text{-WL}) \subseteq \rho(GEL_{k+1}(\Omega, \Theta))$

Proofs[21] rely on connections to logics and techniques from database theory[22,23]

Theorem

if Ω contains concatenation, linear combinations and non-linear activation functions and Θ consists of summation, then $\rho(k\text{-WL}) = \rho(GEL_{k+1}(\Omega, \Theta))$

Approximation properties can be derived as well.

[21] ☞ G., Reutter, Expressiveness and approximation properties of GNNs, ICLR (2022)

[22] ☞ Hella, Libkin, Nurmonen, Wong: Logics with Aggregates, JACM, 48(4): 880-907 (2001)

[23] ☞ Cai, Fürer, Immerman: An optimal lower bound on the number of variables for graph identification. Comb. 12(4):389-410 (1992)

color refinement 1-WL 2-WL k-WL

GraphSage GINs GCNs SGNs

GATs GatedGCNs extended GINs 2-IGNs ChebNet ...

Walk GNNs 2WL-GNNs ring-GNNs 1-Dropout GNNs

Id-aware GNNs CayleyNet 3-IGNs 2-FGNNs ...

kWL-GNNs k-FGNNs (k+1)-IGNs GSNs k-Dropout GNNs ...

6. What's Next?

1. Impact of different aggregation functions on expressive power.

- In initial work is investigated when summation MPNNs can be approximated by mean or max MPNNs, and vice versa.[24]

2. Quantitative approximation results.

- What is complexity of embeddings needed to approximate within ϵ?

[24] ☞ Rosenbluth, Tönshoff and Grohe: Some Might Say All You Need Is Sum, arxiv 2023.

1. Impact of different aggregation functions on expressive power.
   - In initial work is investigated when summation MPNNs can be approximated by mean or max MPNNs, and vice versa.[24]
2. Quantitative approximation results.
   - What is complexity of embeddings needed to approximate within ϵ?

[24] ☞ Rosenbluth, Tönshoff and Grohe: Some Might Say All You Need Is Sum, arxiv 2023.

Technique presented only gives upper bounds. Lower bounds, still case by case analysis.

3. Can we find "reductions" between embedding methods that preserve expressive power.
   - This may help to show lower bounds by simulating "hard" embedding methods.

4. Fining the minimal k in $\mathrm{GEL}_k(\Omega, \Theta)$ needed for your method.
   - The lower k the better the upper bound.
   - (Semantic) treewidth notion for GEL expressions?[25]
   - Related to work on FAQ-AIs when functions and aggregations can be seen as semiring operators.[26]
   - We need to accommodate for non-linear activation functions.

[25] ☞ G* and Reutter. Expressiveness and Approximation Properties of GNNs. ICLR 2022

[26] ☞ Khamis, Ngo and Rudra: "FAQ: Questions Asked Frequently", PODS 2016

Technique presented only gives upper bounds. Lower bounds, still case by case analysis.

3. Can we find "reductions" between embedding methods that preserve expressive power.

- This may help to show lower bounds by simulating "hard" embedding methods.

4. Fining the minimal k in $\mathrm{GEL}_k(\Omega, \Theta)$ needed for your method.

- The lower k the better the upper bound.
- (Semantic) treewidth notion for GEL expressions?[25]
- Related to work on FAQ-AIs when functions and aggregations can be seen as semiring operators.[26]
- We need to accommodate for non-linear activation functions.

[25] ☞ G* and Reutter. Expressiveness and Approximation Properties of GNNs. ICLR 2022

[26] ☞ Khamis, Ngo and Rudra: "FAQ: Questions Asked Frequently", PODS 2016

5. Other hierarchies than Weisfeiler-Leman hierarchy.

- Some embedding methods can be cast in $\mathsf{GEL}_k(\Omega,\Theta)$ but are not as expressive as $(k-1)$-WL.
- E.g., Reconstruction GNNs, ESANs, ID-Aware GNNs, Nested GNNs, Dropout-GNNs.
- By imposing further restrictions on expressions in $\mathsf{GEL}_k(\Omega,\Theta)$ a more fine-grained hierarchy can be obtained. [27], [28]
- More work needed is to understand corresponding separation power. For example, hom count characterizations.

[27] ☞ Qian, Rattan, G*, Morris and Niepert. Ordered Subgraph Aggregation Networks. NeurIPS 2022

[28] ☞ Rattan and Seppelt: Weisfeiler-Leman and Graph Spectra. SODA 2023

6. We can see embedding methods as views.

- Query rewriting using such views?
- View embedding: First embed graph using complex fixed embedding, followed by simple learnable embedding of the view. [29]
- Other?

[29] ☞ Barcel'o, G*, Reutter, Ryschkov: Graph Neural Networks with Local Graph Parameters. NeurIPS 2021

- Generalization properties (VC dimension, Rademacher complexity)
- Semi-ring valued embeddings and learning?
- Zero-one laws of embeddings [30]
- Continuous WL [31]
- More connections with logic and descriptive complexity.[32]

[30] ☞ Adam-Day, Iliant and Ceylan: Zero-One Laws of Graph Neural Networks, arxiv 2023

[31] ☞ Böker, Levie, Huang, Villar and Morris: Fine-grained Expressivity of Graph Neural Networks, arxiv 2023

[32] ☞ Grohe.The Descriptive Complexity of Graph Neural Networks, arxiv 2023

And of course, as also mentioned in Jure's keynote:

Relational embeddings.

- Initial work by considering multi-relation graphs and analyzing power.[33]

[33] ☞ Barceló, Galkin, Morris, Orth: Weisfeiler and Leman Go Relational Learning on Graphs (LoG) 2022.

Thanks to

- The $\mathsf{GEL}(\Omega, \Theta)$ language and use for graph embeddings is heavily influenced by earlier work on the expressive power of linear algebra and matrix query languages together with Robert Brijder, Thomas Muñoz, Cristian Riveros, Jan Van den Bussche, and Domagoj Vrgoč.
- Transferral to ML: Pablo Barceló, Martin Grohe, Christopher Morris, Gaurav Rattan, Juan Reutter, Jasper Steegmans an Jan Van den Bussche.

- There is interest in ML community for these kind of theoretical analyses (but preferably accompanied with some experiments).
- Great opportunity for our community to contribute.
- So far, these papers are in ML conference. Would be great to also have some at PODS or ICDT!
- Maybe you got some inspiration for doing so :-)