Graph Neural Networks: A Primer

Athindran Ramesh Kumar

Princeton University
arkumar@princeton.edu

April 20, 2021

Table of Contents

Motivation for Graph neural nets

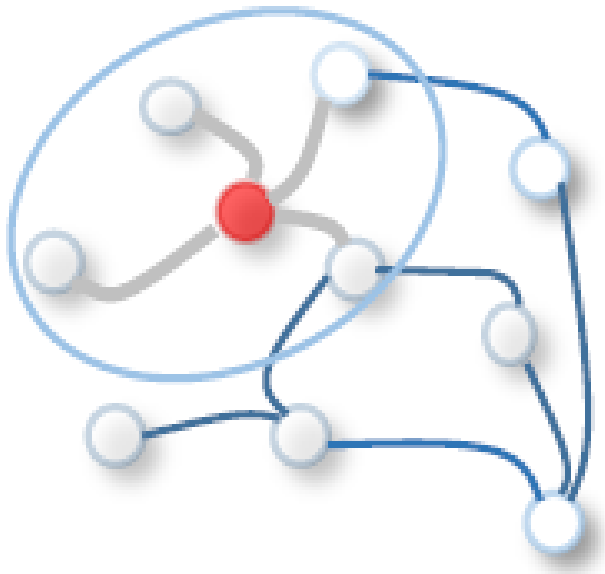
- Want to capture the graph structure in learning
- Convolutional nets are translation invariant
- Graph neural networks needed to capture invariances specific to graphs
- A - the adjacency matrix of the graph
- Naive idea: Vectorize A and use a feed-forward net
- $P^T AP$ for any permutation matrix P is essentially the same graph
- Invariance: $f(P^T AP) = f(A)$, Equivariance:$f(P^T AP) = P^T f(A)P$

Overview

Inference on graphs by using node adjacency information
General purpose learning model - node classification, node regression, graph classification, edge classification ...

Taxonomy

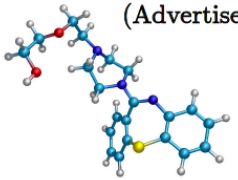
1. Recurrent GNN
2. Convolutional GNN
3. Graph autoencoder
4. Spatio-temporal GNN



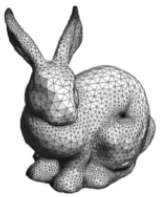
Introduced in Gori, Monfardini, and Scarselli 2005

Applications

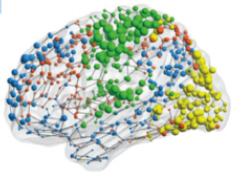
Social networks
(Advertisement)



Drug/Material
molecules
(Chemistry)

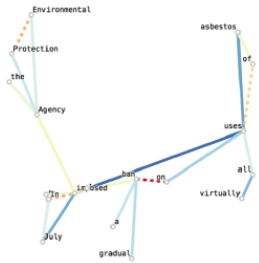


3D Meshes
(Computer Graphics)

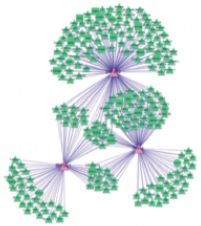


Brain
connectivity
(Neuroscience)

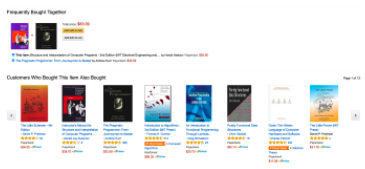
Transportation
networks



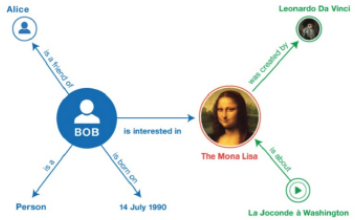
Words relationships
(NLP)



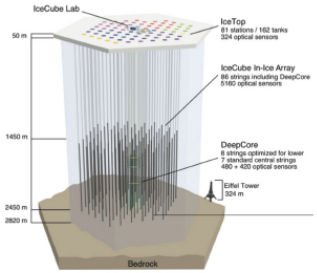
Gene Regulatory
Network



Recommender
systems (Amazon,
Netflix)

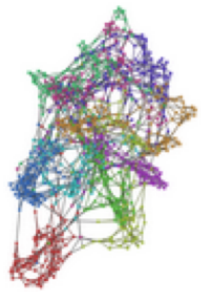


Knowledge graph
(Causality)



Neutrino
detection (High-
energy Physics)

=



Graphs/
Networks

Recurrent graph neural networks

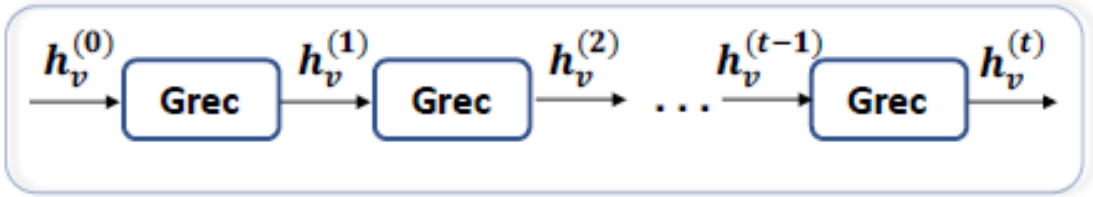
Update node states by exchanging neighborhood information till equilibrium (Scarselli, Gori, et al. 2008, Dai et al. 2018)

$$h_v^t = \sum_{u \in \mathcal{N}(v)} f(h_u^{t-1}, x_u, x_v, x_{uv}; \Theta) \tag{1}$$

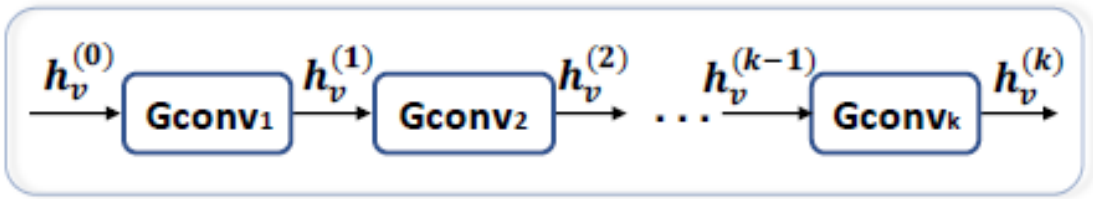
- x_u - Node feature vector
- h_u^t - Node state at time t
- x_{uv} - Edge feature vector
- Θ - Learnable parameters
- Nodes and edges can have additional labels or indices which can optionally be brought into the update.

Recursive connections gaining prominence outside the graph domain too
Questions - stability, restrictions on f

Convolutional graph neural networks



(a) Recurrent Graph Neural Networks (RecGNNs). RecGNNs use the same graph recurrent layer (Grec) in updating node representations.



(b) Convolutional Graph Neural Networks (ConvGNNs). ConvGNNs use a different graph convolutional layer (Gconv) in updating node representations.

Fig. 3: RecGNNs v.s. ConvGNNs

- Broadly divided into spectral ConvGNNs and spatial ConvGNNs
- Unified by GCN (Kipf and Welling 2016) and NN4G (Micheli 2009)

Spectral ConvGNN

- Origins in graph signal processing
- A - adjacency matrix, D - degree matrix, L- Laplacian

$$L = I_n - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \text{ (Laplacian)}$$
$$L = U \Lambda U^T$$
$$H_{:,j}^{(k)} = \sigma \left(\sum_{i=1}^{f_k - 1} U \Theta_{i,j}^k U^T H_{:,i}^{(k-1)} \right)$$

- $H^{(k-1)} \in \mathbb{R}^{n \times f_{k-1}}$, $U \in \mathbb{R}^{n \times n}$
- f_k - No. of channels at layer k
- $\Theta_{i,j}^k \in \mathbb{R}^{n \times n}$
- Compute eigenvectors, fixed graph, 1D features.

Spatial ConvGNN

$$h_v^k = f\left(W^{(k)^T} x_v + \sum_{i=1}^{k-1} \sum_{u \in N(v)} \Theta^{(k)^T} h_u^{(k-1)}\right)$$

- Scores of architectures
- Flexible, efficient, general

Graph Convolutional Network (GCN) - Kipf and Welling 2016

- Spectral ConvGNN - First order approximation

$$H = X *_G g_\Theta = f((I_n - D^{-\frac{1}{2}} A D^{-\frac{1}{2}})X\Theta)$$

- Equivalent to spatial ConvGNN

$$h_v = f\left(\Theta^T \left(\sum_{u \in N(v) \cup v} \bar{A}_{v,u} x_u\right)\right)$$

Graph Convolutional Network (GCN)

First-order approximation

$$X *_G g_\Theta \approx \theta_0' X + \theta_1' (L - I_n) X = \theta_0' X - \theta_1' D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X$$

Constrain $\theta_1 = -\theta_0$

$$X *_G g_\Theta \approx \theta_0' (I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) X$$

This is an unstable operator. Re-normalize with $\bar{A} = A + I_n$ and corresponding degree matrix $\bar{D}$

$$Z = \bar{D}^{-\frac{1}{2}} \bar{A} \bar{D}^{-\frac{1}{2}} X \Theta$$

Other networks

Graph Autoencoders - Unsupervised learning (Cao, Lu, and Q. Xu 2016; Wang, Cui, and Zhu 2016)

Two major functions:

1. Network/node embedding - Low-level representation used to reconstruct graph properties
2. Graph generation - Generative model to generate graphs from embedding

Application : Molecular drug discovery

Spatiotemporal networks (J. Zhang et al. 2018; Li et al. 2017)

- Real-world applications with dynamic graph structure and features
- Opinions in a social network, road traffic
- Optional convolution over time with tensor input

Table of Contents

Universal Approximation for Node Regression

Graph neural networks are universal approximators - Scarselli, Gori, et al. 2008

- G an undirected graph, n a particular node
- $ne[n]$ neighbors of node, $co[n]$ edges of node, h_n - node state, x_n - node input
- GNN mapping $\psi(G, n) \in \mathbb{R}^m$
- $\mathcal{L} = \{G_i, n_{ij}, t_{ij}\}$ - set of graphs with nodes and targets for each node
- Node regression - minimize squared error over all targets and predictions

$$h_n = f_w(x_n, x_{co[n]}, h_{ne[n]}, x_{ne[n]}) \tag{2}$$
$$o_n = g_w(h_n, x_n) \tag{3}$$

Entire graph:

$$h = F_w(h, x)$$
$$o = G_w(h, x)$$

Universal Approximation for Node Regression

Unfolding equivalence

Two nodes are unfolding equivalent if their unfolding trees are the same

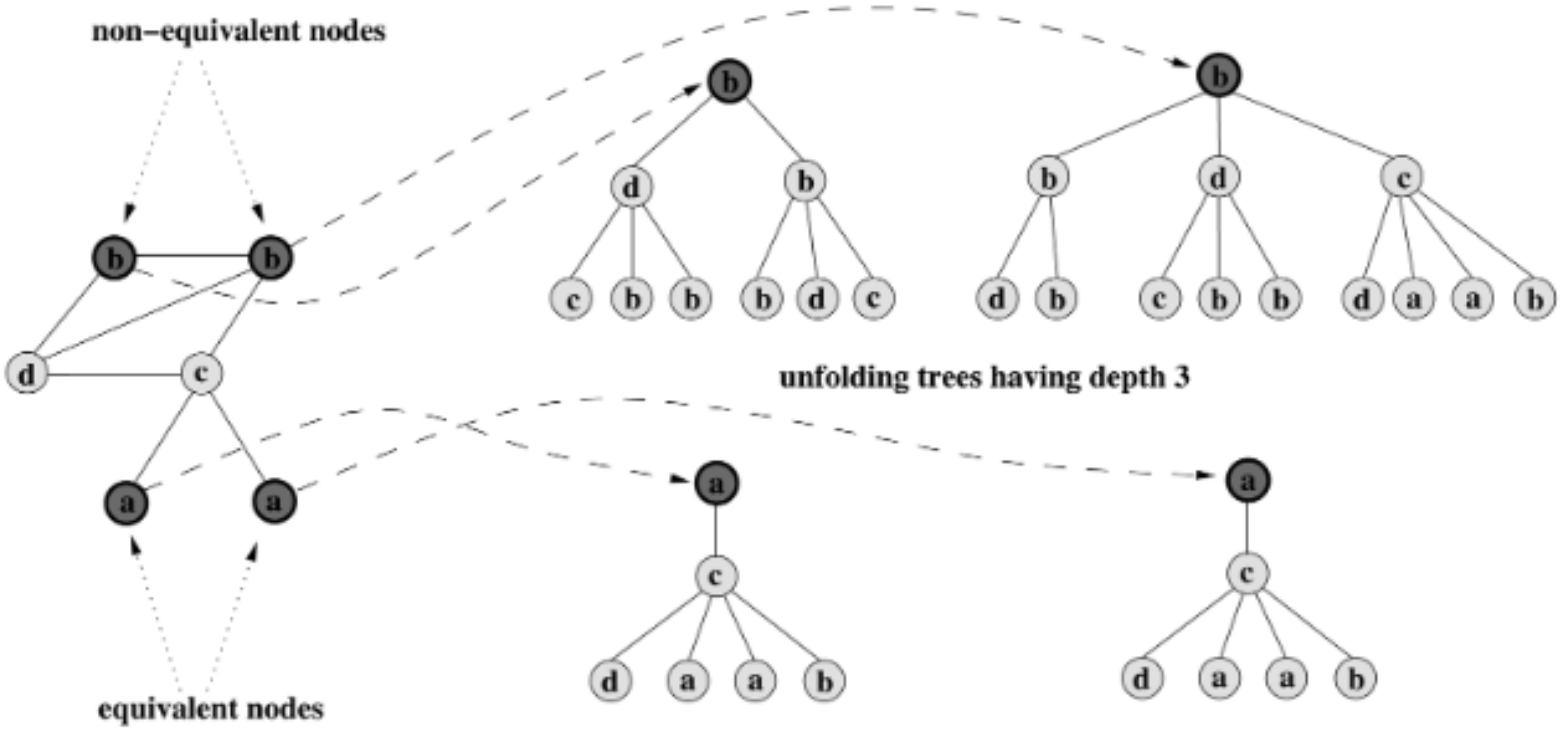


Fig. 2. Graph and four unfolding trees of depth 3. Dashed lines specify the correspondence between a node and its unfolding tree. The two nodes not unfolding equivalent because their unfolding trees are different, whereas the two nodes with label a are unfolding equivalent.

Universal Approximation - Key Results

$\mathcal{D}$ - space of graphs and nodes in the graph
Functions preserving unfolding equivalence - $\mathcal{F}(\mathcal{D})$

Theorem

A function ℓ belongs to $\mathcal{F}(\mathcal{D})$ if and only if there exists κ defined on trees such that $\ell(G, n) = \kappa(T_n)$ for any node n of the domain $\mathcal{D}$

Define a probability measure on $\mathcal{D}$

Theorem

For any measurable function $\tau \in \mathcal{F}(\mathcal{D})$ that preserves unfolding equivalence, any norm $\|\|\|$ on $\mathcal{R}^m$, any probability measure P on $\mathcal{D}$, and any reals $\epsilon > 0$, $0 < \mu < 1$, $0 < \lambda < 1$, there exists two continuosly differentiable functions f and g such that the global transition function is a contraction map with a contracting constant μ, the stable state is uniformly bounded and the corresponding map ψ satisfies:

$$P(\|\tau(G, n) - \psi(G, n)\| \geq \epsilon) \leq 1 - \lambda$$

Universal Approximation - Proof Technique

- Divide the space $\mathcal{D}$ into graphs with similar structure and further have a fine-grained division into hypercubes on the space of target of each node in the graph
- Pick one graph-target from each hypercube as a representative set to approximate. Can show that approximating this set is sufficient to approximate the entire space
- There exists an injective map from the space of unfolding trees to an integer number. Choose f to be this injective map
- Show that GNN can implement this map while ensuring stability
- Choose $g = \kappa(f^{-1}(\cdot))$

Table of Contents

Graph Classification

Specific problem of differentiation graph structures K. Xu et al. 2018

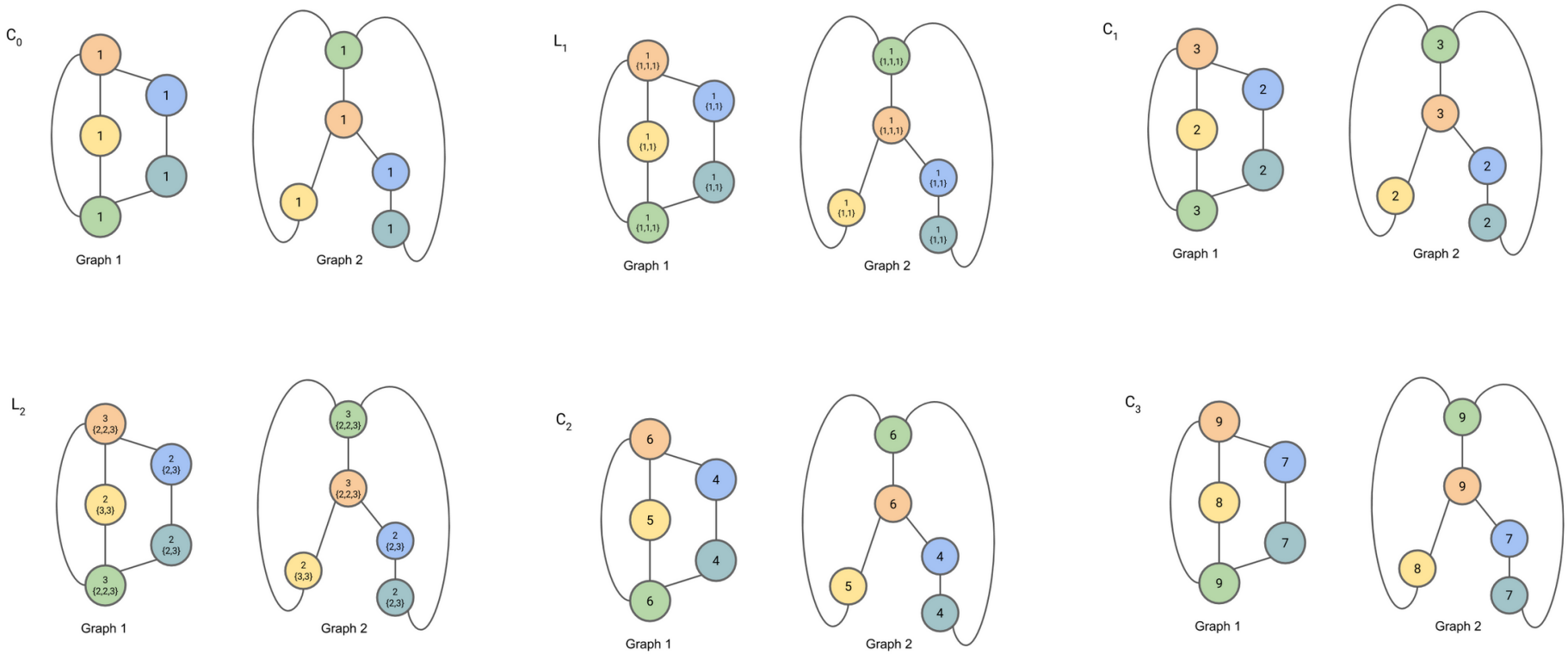
- Multiset - A collection of elements where order is not important but I keep track of the count of each element

Weisfeiler-Lehman test Leman and Weisfeiler 1968

- Initialize $C_{0,n} = \text{MultiSet}(\{1\})$ for all nodes
- Iteratively set $C_{t,n} = \text{Compress}\left(\cup_{m \in Ne[n]}(C_{t-1,n} \cup C_{t-1,m})\right)$
- Compression has to be consistent and injective
- Can optionally iterate until convergence
- After k iterations, if the node labels differ, graphs are not isomorphic
- Otherwise, inconclusive

Graph Classification

Weisfeiler-Lehman test



Distinguishing graph structure - unknown whether its P or NP.
Are graph neural networks as powerful as Weisfeiler-Lehman test?

Distinguishing graph structure

Convolutional architecture under consideration:

$$a_v^k = \text{AGGREGATE}^k\left(\{h_u^{k-1} : u \in Ne[v]\}\right), \; h_v^k = \text{COMBINE}^k\left(h_v^{k-1}, a_v^k\right)$$

Some heuristic choices:

1. GraphSAGE (Hamilton, Ying, and Leskovec 2017):

$$a_v^k = \text{MAX}\left(\{\text{ReLU}\left(W \cdot h_u^{k-1}\right), u \in Ne[v]\}\right), \; h_v^k = W \cdot [h_v^{k-1}, a_v^k]$$

2. GCN (Kipf and Welling 2016):

$$h_v^k = \text{ReLU}\left(W.\text{MEAN}\{h_u^{k-1}, \forall u \in Ne[v] \cup v\}\right)$$

For graph representation,

$$h_G = \text{READOUT}(\{h_v^K | v \in G\})$$

Distinguishing graph structure

Definition

A multiset is a generalized concept of a set that allows multiple instances for its elements. Multiset is $X = (S, m)$ where S is the underlying set that is formed from its distinct elements and $m : S \to \mathbb{N}_{\geq 1}$ gives the multiplicity of its elements.

- Represent features on each node as countable sets
- Maximal unfolding tree of each node represented by a multi-set of features
- **Key idea**: GNN is maximally powerful if the mapping from the neighborhood multiset to the representation is injective

Key results:

- None of the graph neural networks here are more powerful than WL test
- GNN however learns an embedding unlike WL
- With a sufficient number of layers, GNN as powerful if the AGGREGATE, COMBINE and READOUT functions are injective on the domain of multisets

Distinguishing graph structure

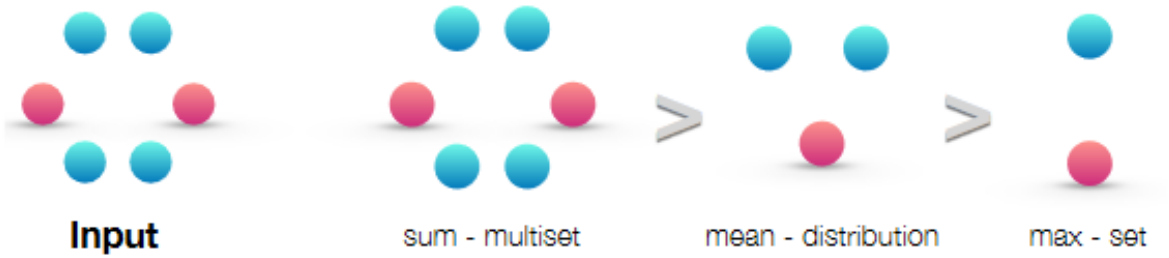
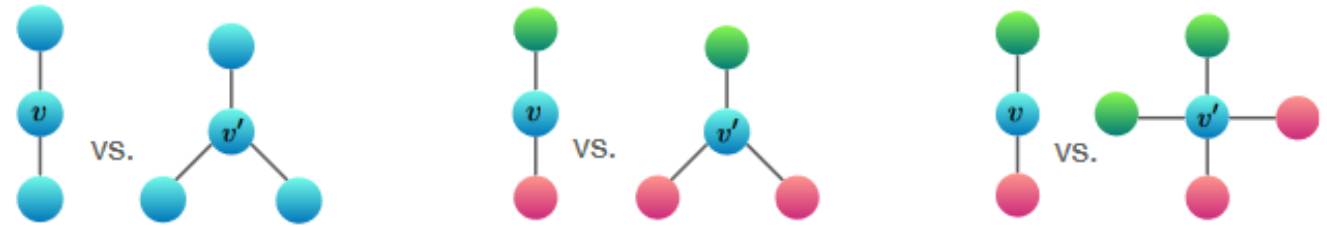


Figure 2: **Ranking by expressive power for sum, mean and max aggregators over a multiset.** Left panel shows the input multiset, *i.e.*, the network neighborhood to be aggregated. The next three panels illustrate the aspects of the multiset a given aggregator is able to capture: sum captures the full multiset, mean captures the proportion/distribution of elements of a given type, and the max aggregator ignores multiplicities (reduces the multiset to a simple set).



(a) Mean and Max both fail (b) Max fails (c) Mean and Max both fail

Figure 3: **Examples of graph structures that mean and max aggregators fail to distinguish.** Between the two graphs, nodes v and v' get the same embedding even though their corresponding graph structures differ. Figure 2 gives reasoning about how different aggregators "compress" different multisets and thus fail to distinguish them.

Both graphSAGE and GCN flawed in their architecture

Distinguishing graph structure

Lemma

Assume $\mathcal{X}$ is countable. There exists a function $f : \mathcal{X} \to \mathbb{R}^n$ so that $h(X) = \sum_{x \in X} f(x)$ is unique for each multiset $X \subset \mathcal{X}$ of bounded size. Moreover, any multiset function g can be decomposed as $\phi\left(\sum_{x \in X} f(x)\right)$ for some ϕ.

Simple observation gives rise to GIN (Graph isomorphism network):

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

Some empirical results validate the theoretical findings

Moral of the story: Choose the structure that is right for the task at hand. If the downstream task wants to differentiate graph structure, GIN better than GCN and graphSAGE

Table of Contents

Graph isomorphism testing and approximation

Notation and Definitions

- Graph G represented as $\mathcal{X}^{n\times n}$. $\mathcal{X}$ - compact set. n number of nodes. The diagonal elements are node labels and off-diagonal elements are edge weights

Definition

Let $\mathcal{C}$ be a collection of permutation-invariant functions from $\mathcal{X}^{n\times n} \to \mathbb{R}$. We say $\mathcal{C}$ is GIso-discriminating if for all non-isomorphic $G_1, G_2 \in \mathcal{C}^{n\times n}$, there exists a function $h \in \mathcal{C}$ such that $h(G_1) \neq h(G_2)$

Definition

Let $\mathcal{C}$ be a collection of permutation-invariant functions from $\mathcal{X}^{n\times n} \to \mathbb{R}$. We say $\mathcal{C}$ is universally approximating if for all permutation-invariant function f from $\mathcal{X}^{n\times n} \to \mathbb{R}$, and for all $\epsilon > 0$, there exists $h_{f,\epsilon} \in \mathcal{C}$ such that $\|f - h_{f,\epsilon}\|_\infty = \sup_{G\in\mathcal{X}_{n\times n}} |f(G) - h_{f,\epsilon}(G)| < \epsilon$

Graph isomorphism testing and approximation

Theorem

If $\mathcal{C}$ is universally approximating, then it is also GIso-discriminating

Proof idea: If the function class is universally approximating, then it can learn a function that indicates whether the graph belongs to a particular equivalence class (graphs with same isomorphism)

Theorem

If $\mathcal{C}$, a collection of continuous permutation-invariant functions from $\mathcal{X}^{n\times n} \to \mathbb{R}$, is GIso-discriminating, then C^{+3} is universally approximating.

Proof idea: If a function class is isomorphism discriminating, then adding three more layers to the function class is sufficient for universal approximation. First layer, zero out all graphs with different isomorphism. Second layer, make an indicator function on each equivalence class. Third layer, map the required function. Note that the input space is assumed to be finite. Can extend to continuous space too using measure theoretic notions.

Table of Contents

Learning graph moments

- A - adjacency matrix of graph G
- n - number of nodes in graph G

$$M_p(A) = \prod_{q=1}^{p} (A \cdot W_q + B_q)$$

is p^{th} order graph moment with W_q, B_q being $n \times n$ matrices. For node permutation invariance,

$$W, B = cI, \text{ or } W, B = c11^{\mathsf{T}}$$

Can encode topological properties of the graph e.g degree, paths of given length. Provides information about the graph generation process

Learn a functional approximator: $F : A \to M_p(A)$

GCN's vs feedforward networks

Class of GCNs considered here is more general. A single layer:

$$F(A, x) = \sigma(f(A) \cdot x \cdot W + b)$$

where x_i is the attribute of node i

Encompasses graphSAGE (Hamilton, Ying, and Leskovec 2017) and GCN (Kipf and Welling 2016) considered before

Fully connected networks vectorize the adjacency matrix so clearly inferior. Formally:

Theorem

A fully connected network with one hidden layer requires $n > O(C_f^2) \approx O(p^2 N^{2q})$ number of neurons in the best case with $1 \leq q \leq 2$ to learn a graph moment of order p for graphs with N nodes. It also needs $S > O(nd) \approx O(p^2 N^{2q+2})$ number of samples to make the learning tractable

GCN's vs feedforward networks

Theorem

With the number of layers n greater or equal to the order p of a graph moment $M_p(A)$, graph convolutional networks with residual connections can learn a graph moment M_p with $O(p)$ number of neurons, independent of the size of the graph.

With less than p layers, order p graph moment cannot be learnt by a GCN

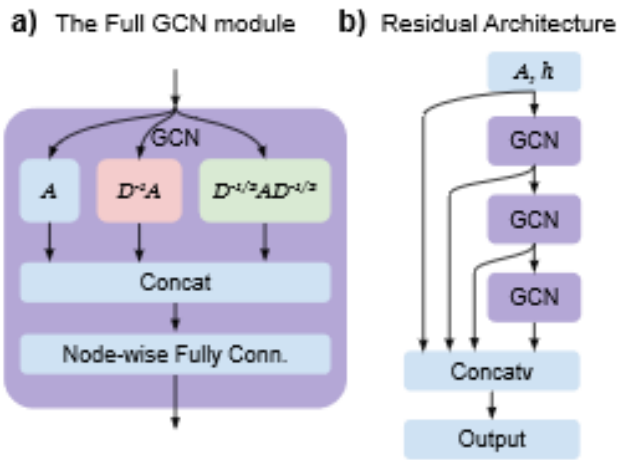


Figure 3: GCN layer (a), using three different propagation rules and a node-wise FC layer. Using residual connections (b) allows a n-layer modular GCN to learn any polynomial function of order n of its constituent operators.

Table of Contents

Linear invariant and equivariant layers

Maron, Ben-Hamu, Shamir, et al. 2018

- Adjacency matrix: $\mathbb{R}^{n \times n}$
- Generalize to hyper-edges: form a set of k nodes, attribute a weight to hyper-edge
- Resulting tensor $\mathbb{R}^{n^k}$
- Invariance: $f(P^T A P) = f(A)$ for a permutation matrix P
- Equivariance: $f(P^T A P) = P^T f(A) P$ for a permutation matrix P
- Let $L : \mathbb{R}^{n^k} \to \mathbb{R}$ be a linear operator. Under what conditions will L be invariant?
- Let $L : \mathbb{R}^{n^k} \to \mathbb{R}^{n^k}$ be a linear operator. Under what conditions will L be equivariant?

Linear invariant and equivariant layers

Invariance:
For $L : \mathbb{R}^{n \times n} \to \mathbb{R}$, matrix $L \in \mathbb{R}^{1 \times n^2}$. For permutation matrix P,
$L\text{vec}(P^T A P) = L\text{vec}(A)$
Use the property, $\text{vec}(XAY) = Y^T \otimes X\text{vec}(A)$
Reduced to: $P \otimes P\text{vec}(L) = \text{vec}(L)$
Equivariance:
For $L : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}$, matrix $L \in \mathbb{R}^{n^2 \times n^2}$
$[L\text{vec}(P^T A P)] = P^T [L\text{vec}(A)] P$
Reduced to: $P \otimes P \otimes P \otimes P\text{vec}(L) = \text{vec}(L)$

For general case:

$$\text{invariant } L : P^{\otimes k}\text{vec}(L) = \text{vec}(L)$$

$$\text{equivariant } L : P^{\otimes 2k}\text{vec}(L) = \text{vec}(L)$$

Solving the fixed point equations

$$P^{\otimes \ell} \mathrm{vec}(X) = \mathrm{vec}(Q * X)$$

for a permutation matrix Q.

- Define a relation on the index space of tensors in $\mathbb{R}^{n^\ell}$
- For multi-indices $\mathsf{a}, \mathsf{b} \in [n]^\ell$, we set $\mathsf{a} \sim \mathsf{b}$ if $\mathsf{a}_i = \mathsf{a}_j$ implies $\mathsf{b}_i = \mathsf{b}_j$
- For $n = 2, \ell = 2$, example, two equivalence classes $\{(1,1),(2,2)\}$ and $\{(1,2),(2,1)\}$
- For each equivalence class $\gamma \in [n]^\ell / \sim$, we define an order-$\ell$ tensor $B^\gamma \in \mathbb{R}^{n^\ell}$

$$B^\gamma_{\mathsf{a}} = \begin{cases} 1 & \mathsf{a} \in \gamma \\ 0 & \text{otherwise} \end{cases}$$

- For $n = 2, l = 2$, example,

$$B^{\gamma_1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, B^{\gamma_2} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Solving the fixed point equations

- The vectors B^γ form a basis for the solution set of the fixed point equation
- In other words, any linear operator L has to have equal entries on the equivalence classes

Theorem

The space of invariant (equivariant) linear layers $\mathbb{R}^{n^k} \to \mathbb{R}(\mathbb{R}^{n^k} \to \mathbb{R}^{n^k})$ *is of dimension* $b(k)(b(2k))$ *with basis elements* B^γ*, where* γ *are equivalence classes in* $[n]^k/\sim$ $([n]^{2k}/\sim)$.

The results can be extended to linear layer with biases, vector features and mixed-order layers.
$b(k)$ - k^{th} bell number, count the number of possible partitions of a set with k elements

- G-invariant network - Use linear equivariant and invariant layers with element-wise activation
- How expressive are these networks?

Universality of Invariant Networks

Maron, Fetaya, et al. 2019

Theorem

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous G-invariant function for some $G \leq S_n$, and $K \subset \mathbb{R}^n$ a compact set.There exists a G-invariant network that approximates f to an arbitrary precision.

S_n- symmetric group (Bijection from $\{1 \dots n\}$ to itself)
Vague statement... The constructed network has at least 3 equivariant layers
Bad news, the tensor order k scales as $\frac{n(n-1)}{2}$ - Reason: there exists graphs which need higher-order information to be able to differentiate between them. Very difficult to keep up with the expanding memory and compute requirements on large-scale graphs

Back to isomorphism testing...

- We saw first-order graph neural networks are as strong as Weisfeiler-Lehman for structure differentiation
- Is there any benefit in including k^{th} order hyper-edges? Maron, Ben-Hamu, Serviansky, et al. 2019

k^{th} Weisfeiler-Lehman test

- $G = (V, E, d)$ be a colored graph where $|V| = n$ and $d : V \to \Sigma$ where Σ is set of colors
- k-WL constructs a coloring for k-tuple of vertices: $c : V^k \to \Sigma$
- Tensor $C \in \mathbb{R}^{n^k}$ represents color of all k-tuples
- Initial coloring should be consistent.. isomorphic k-tuples get same color

$$\mathrm{I} = (i_1, i_2, \ldots, i_k) \tag{4}$$

$$N_j(\mathrm{I}) = \{(i_1, \ldots, i_{j-1}, i', i_{j+1}, \ldots, i_k) | i' \in [n]\} \tag{5}$$

$$C_{\mathrm{I}}^{\ell} = \mathrm{enc}\left(C_{\mathrm{I}}^{\ell-1}, \left(\{C_{\mathrm{J}}^{\ell-1} | \mathrm{J} \in N_j(I)\} \middle| j \in [k]\right)\right) \tag{6}$$

Power sum symmetric polynomials

For each $k \geq 2$ there is a pair of non-isomorphic graphs distinguishable by $(k+1)WL$ but not by $(k)WL$.

k-order networks

- Represent colors as vectors.. Tensor $C \in \mathbb{R}^{n^k \times a}$
- Multiset representation $X \in \mathbb{R}^{n \times a}$ that is invariant to permutations of nodes (rows)
- Let $\alpha = (\alpha_1, \alpha_1, \ldots, \alpha_a) \in [n]^a$ be a multi-index and for $y \in \mathbb{R}^a$, set

$$y^\alpha = y_1^{\alpha_1} \cdot y_2^{\alpha_2} \ldots y_a^{\alpha_a}$$

- Represent $X = [x_1, x_2, \ldots x_n]^T$

$$p_\alpha(X) = \sum_{i=1}^{n} x_i^\alpha$$

- For $\sum_{j=1}^{a} \alpha_j \leq n$, the set p_α can be used to compose the polynomials which respect permutation symmetry

k-order Graph Networks

- Ring of multi-symmetric polynomials $q(X) = q(g \cdot X)$ can be represented as $r(u(X))$ where:

$$u(x) = (p_\alpha(X) \Big| |\alpha| \leq n)$$

  with arbitrary polynomial r

- Proposition: $u(X)$ is an unique representation of X
- **Key idea**: $u(X)$ can be expressed using linear equivariant layers with a MLP to approximate the polynomials

Theorem

Given two graphs $G_1 = (V_1, E_1, d_1), G_2 = (V_2, E_2, d_2)$ that can be distinguished by the k-WL graph isomorphism test, there exists a k-order network F so that $F(G_1) \neq F(G_2)$. On the other direction for every two isomorphic graphs $G_1 \sim G_2$ and k-order network F, $F(G_1) = F(G_2)$.

Table of Contents

Traditional generalization theory

- Targets Y, inputs X from distribution $\mathcal{D}$, predictions $\hat{Y}(X)$
- Loss function $\mathcal{L}(Y, \hat{Y}(X))$
- Typical generalization bound: w.p $1 - \delta$

$$\mathbb{E}_{X \sim \mathcal{D}}[\mathcal{L}(Y, \hat{Y}(X))] = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(Y, \hat{Y}(X_i)) + \triangle(m, \delta, \chi) \tag{7}$$

$$\triangle \propto \sqrt{\frac{1}{m}}$$

$\triangle \propto \chi$ (Some measure of model complexity)

Popular complexity measures: VC-dimension, Rademacher complexity.
Another approach to generalization called PAC-Bayes

Generalization for graph networks

- VC-dimension (Scarselli, Tsoi, and Hagenbuchner 2018)
- Rademacher (Garg, Jegelka, and Jaakkola 2020)
- Algorithmic stability (Verma and Z.-L. Zhang 2019)
- PAC-Bayes bounds (Liao, Urtasun, and Zemel 2020))

Statistics	Max Node Degree $d-1$	Max Hidden Dim h	Spectral Norm of Learned Weights
VC-Dimension (Scarselli et al., 2018)	-	$\mathcal{O}(h^4)$	-
Rademacher Complexity (Garg et al., 2020)	$\mathcal{O}\left(d^{l-1}\sqrt{\log(d^{2l-3})}\right)$	$\mathcal{O}(h\sqrt{\log h})$	$\mathcal{O}\left(\lambda\mathcal{C}\xi\sqrt{\log(\|W_2\|_2\lambda\xi^2)}\right)$
Ours	$\mathcal{O}(d^{l-1})$	$\mathcal{O}(\sqrt{h\log h})$	$\mathcal{O}\left(\lambda^{1+\frac{1}{l}}\xi^{1+\frac{1}{l}}\sqrt{\|W_1\|_F^2+\|W_2\|_F^2+\|W_l\|_F^2}\right)$

Table 1: Comparison of generalization bounds for GNNs. "-" means inapplicable. l is the network depth. Here $\mathcal{C} = C_\phi C_\rho C_g \|W_2\|_2$, $\xi = C_\phi \frac{(d\mathcal{C})^{l-1}-1}{d\mathcal{C}-1}$, $\zeta = \min(\|W_1\|_2, \|W_2\|_2, \|W_l\|_2)$, and $\lambda = \|W_1\|_2\|W_l\|_2$. More details about the comparison can be found in Appendix A.5.

Figure: PAC-Bayes bounds

Benign over-fitting largely not studied

Conclusion

- Graph neural networks capture structure and invariances specific to graphs
- Universal approximation of some class of invariant functions
- Relationship between isomorphism testing and universal approximation
- Including higher order tensors leads to increased power at the expense of computation and memory

References I

Cao, Shaosheng, Wei Lu, and Qiongkai Xu (2016). "Deep neural networks for learning graph representations". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1.

Dai, Hanjun et al. (2018). "Learning steady-states of iterative algorithms over graphs". In: *International conference on machine learning*. PMLR, pp. 1106–1114.

Garg, Vikas, Stefanie Jegelka, and Tommi Jaakkola (2020). "Generalization and representational limits of graph neural networks". In: *International Conference on Machine Learning*. PMLR, pp. 3419–3430.

Gori, Marco, Gabriele Monfardini, and Franco Scarselli (2005). "A new model for learning in graph domains". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 2. IEEE, pp. 729–734.

Hamilton, William L, Rex Ying, and Jure Leskovec (2017). "Inductive representation learning on large graphs". In: *arXiv preprint arXiv:1706.02216*.

Kipf, Thomas N and Max Welling (2016). "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907*.

References II

Leman, AA and B Weisfeiler (1968). "A reduction of a graph to a canonical form and an algebra arising during this reduction". In: *Nauchno-Technicheskaya Informatsiya* 2.9, pp. 12–16.

Li, Yaguang et al. (2017). "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting". In: *arXiv preprint arXiv:1707.01926*.

Liao, Renjie, Raquel Urtasun, and Richard Zemel (2020). "A PAC-Bayesian Approach to Generalization Bounds for Graph Neural Networks". In: *arXiv preprint arXiv:2012.07690*.

Maron, Haggai, Heli Ben-Hamu, Hadar Serviansky, et al. (2019). "Provably powerful graph networks". In: *arXiv preprint arXiv:1905.11136*.

Maron, Haggai, Heli Ben-Hamu, Nadav Shamir, et al. (2018). "Invariant and equivariant graph networks". In: *arXiv preprint arXiv:1812.09902*.

Maron, Haggai, Ethan Fetaya, et al. (2019). "On the universality of invariant networks". In: *International conference on machine learning*. PMLR, pp. 4363–4371.

Micheli, Alessio (2009). "Neural network for graphs: A contextual constructive approach". In: *IEEE Transactions on Neural Networks* 20.3, pp. 498–511.

References III

- Scarselli, Franco, Marco Gori, et al. (2008). "The graph neural network model". In: *IEEE transactions on neural networks* 20.1, pp. 61–80.
- Scarselli, Franco, Ah Chung Tsoi, and Markus Hagenbuchner (2018). "The Vapnik–Chervonenkis dimension of graph and recursive neural networks". In: *Neural Networks* 108, pp. 248–259.
- Verma, Saurabh and Zhi-Li Zhang (2019). "Stability and generalization of graph convolutional neural networks". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1539–1548.
- Wang, Daixin, Peng Cui, and Wenwu Zhu (2016). "Structural deep network embedding". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1225–1234.
- Xu, Keyulu et al. (2018). "How powerful are graph neural networks?" In: *arXiv preprint arXiv:1810.00826*.
- Zhang, Jiani et al. (2018). "Gaan: Gated attention networks for learning on large and spatiotemporal graphs". In: *arXiv preprint arXiv:1803.07294*.