On the equivalence between graph isomorphism testing and function approximation with GNNs

Zhengdao Chen, Soledad Villar, Lei Chen and Joan Bruna

Courant Institute of Mathematical Sciences
Center for Data Science

Paper to appear at NeurIPS 2019
(https://arxiv.org/abs/1905.12560)
October 27, 2019

How powerful are graph neural networks?

Perspective 1 - In terms of graph isomorphism testing

Q: How good are GNNs at distinguishing non-isomorphic graphs?

Xu, Hu, Leskovec, Jegelka (2019); Weisfeler, Lehman (1968)

How powerful are graph neural networks?

Perspective 1 - In terms of graph isomorphism testing

Q: How good are GNNs at distinguishing non-isomorphic graphs?

For example, Xu et al. (2019) shows that any GNN defined in the following way

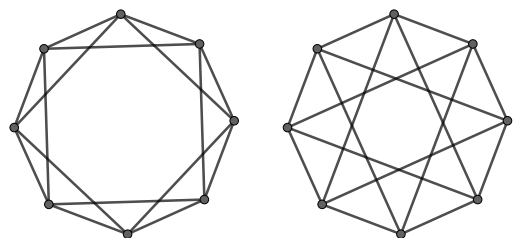
$$a_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\{h_u^{(k-1)} : u \in \mathcal{N}(u)\}\right)$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right)$$

can be at most as good as the Weisfeler-Lehman test.

Xu, Hu, Leskovec, Jegelka (2019); Weisfeler, Lehman (1968)

In particular, they cannot distinguish non-isomorphic regular graphs with the same degree.



How powerful are graph neural networks?

Perspective 2 - In terms of function approximation

Q: How good are GNNs at approximating (invariant) functions on graphs?

How powerful are graph neural networks?

Perspective 2 - In terms of function approximation

Q: How good are GNNs at approximating (invariant) functions on graphs?

Usually, on graph data, we care about invariant or equivariant functions. Intuitively, these are the functions that don't care about the ordering of the nodes.

How powerful are graph neural networks?

Perspective 2 - In terms of function approximation

Q: How good are GNNs at approximating (invariant) functions on graphs?

Usually, on graph data, we care about invariant or equivariant functions. Intuitively, these are the functions that don't care about the ordering of the nodes.

Let's introduce them rigorously.

Invariant functions on graph data

For a graph with n nodes, a permutation on the node set is represented by a permutation matrix π. The total collection of these matrices forms the permutation group, $\mathbb{S}_n$

Invariant functions on graph data

For a graph with n nodes, a permutation on the node set is represented by a permutation matrix π. The total collection of these matrices forms the permutation group, $\mathbb{S}_n$

If graph data is represented by matrices $\in \mathbb{R}^{n\times n}$, then
Definition 1: A function $f : \mathbb{R}^{n\times n} \to \mathbb{R}$ is *invariant* if
$\forall A \in \mathbb{R}^{n\times n}, \forall \pi \in \mathbb{S}_n$, there is $f(\pi^\intercal A \pi) = f(A)$

Invariant functions on graph data

For a graph with n nodes, a permutation on the node set is represented by a permutation matrix π. The total collection of these matrices forms the permutation group, $\mathbb{S}_n$

If graph data is represented by matrices $\in \mathbb{R}^{n \times n}$, then
Definition 1: A function $f : \mathbb{R}^{n \times n} \to \mathbb{R}$ is *invariant* if
$\forall A \in \mathbb{R}^{n \times n}, \forall \pi \in \mathbb{S}_n$, there is $f(\pi^\intercal A \pi) = f(A)$

More generally, if graph data is represented by a k-th order tensors $\in \mathbb{R}^{n^k \times d}$, then
Definition 1 (general): A function $f : \mathbb{R}^{n^k \times d} \to \mathbb{R}$ is *invariant* if
$\forall A \in \mathbb{R}^{n^k \times d}, \forall \pi \in \mathbb{S}_n$, there is $f(\pi * A) = f(A)$, where $\pi * A$ means applying the permutation to all of the k dimensions of A with n entries.

Equivariant functions on graph data

If graph data is represented by matrices $\in \mathbb{R}^{n\times n}$, then
Definition 2: A function $f : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ is *equivariant* if $\forall A \in \mathbb{R}^{n\times n}, \forall \pi \in \mathbb{S}_n$, there is $f(\pi^\intercal A \pi) = \pi^\intercal f(A)\pi$

Equivariant functions on graph data

If graph data is represented by matrices $\in \mathbb{R}^{n \times n}$, then
Definition 2: A function $f : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}$ is *equivariant* if $\forall A \in \mathbb{R}^{n \times n}, \forall \pi \in \mathbb{S}_n$, there is $f(\pi^\intercal A \pi) = \pi^\intercal f(A) \pi$

More generally, if graph data is represented by a k-th order tensors $\in \mathbb{R}^{n^k \times d}$, then
Definition 2 (general): A function $f : \mathbb{R}^{n^k \times d} \to \mathbb{R}^{n^{k'} \times d'}$ is *equivariant* if $\forall A \in \mathbb{R}^{n^k \times d}, \forall \pi \in \mathbb{S}_n$, there is $f(\pi * A) = \pi * f(A)$.

Equivariant functions on graph data

If graph data is represented by matrices $\in \mathbb{R}^{n\times n}$, then
Definition 2: A function $f : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ is *equivariant* if $\forall A \in \mathbb{R}^{n\times n}, \forall \pi \in \mathbb{S}_n$, there is $f(\pi^\intercal A \pi) = \pi^\intercal f(A)\pi$

More generally, if graph data is represented by a k-th order tensors $\in \mathbb{R}^{n^k \times d}$, then
Definition 2 (general): A function $f : \mathbb{R}^{n^k \times d} \to \mathbb{R}^{n^{k'} \times d'}$ is *equivariant* if $\forall A \in \mathbb{R}^{n^k \times d}, \forall \pi \in \mathbb{S}_n$, there is $f(\pi * A) = \pi * f(A)$.

Invariant functions can be regarded as special cases of equivariant functions when $k' = 1$.

Approximating invariant functions with GNNs

GNNs for graph classification can be considered as functions from $\mathbb{R}^{n \times n \times d}$ to $\mathbb{R}$. Thanks to their architecture, most GNNs are invariant functions.

Approximating invariant functions with GNNs

GNNs for graph classification can be considered as functions from $\mathbb{R}^{n \times n \times d}$ to $\mathbb{R}$. Thanks to their architecture, most GNNs are invariant functions.

But, can every ("nice") *invariant* function from $\mathbb{R}^{n \times n \times d}$ to $\mathbb{R}$ be approximated by GNNs, just like every ("nice") general function can be approximated by feedforward neural networks?

Approximating invariant functions with GNNs

GNNs for graph classification can be considered as functions from $\mathbb{R}^{n \times n \times d}$ to $\mathbb{R}$. Thanks to their architecture, most GNNs are invariant functions.

But, can every ("nice") *invariant* function from $\mathbb{R}^{n \times n \times d}$ to $\mathbb{R}$ be approximated by GNNs, just like every ("nice") general function can be approximated by feedforward neural networks?

The answer is not trivial, and depends on the specific GNN architecture.

Example: G-invariant Networks

For example, for the G-invariant Networks proposed by Maron et al. (2019), the answer is yes if we allow the order of the tensors that appear in the hidden layers to grow polynomially in the number of nodes in the graph. But not clear if the tensor order is restricted.

Maron, Fetaya, Segol, Lipman (2019)

Example: G-invariant Networks

For example, for the G-invariant Networks proposed by Maron et al. (2019), the answer is yes if we allow the order of the tensors that appear in the hidden layers to grow polynomially in the number of nodes in the graph. But not clear if the tensor order is restricted.

In fact, we will show later that if the tensor order is fixed to be 2, then they are not universal.

Maron, Fetaya, Segol, Lipman (2019)

Bridging the two perspectives

The first goal of this paper is to show that the two perspectives (distinguishing non-isomorphic graphs and approximating invariant functions universally) are equivalent.

Bridging the two perspectives

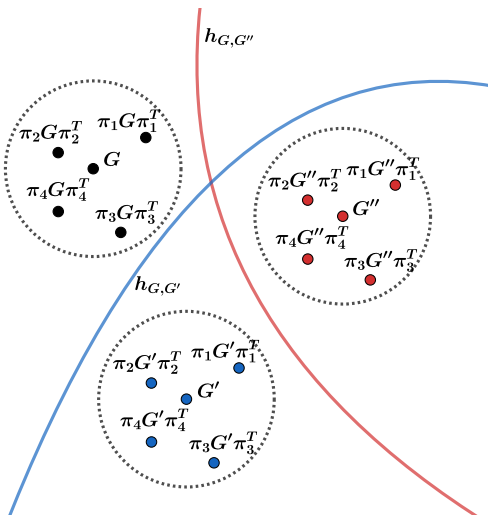
The first goal of this paper is to show that the two perspectives (distinguishing non-isomorphic graphs and approximating invariant functions universally) are equivalent.

To do this, we need to formulate both perspectives rigorously.

Bridging the two perspectives

GIso-discriminating

If $\mathcal{C}$ is a family of invariant functions from $\mathcal{X}^{n\times n}$ to $\mathbb{R}$, we call it *GIso-discriminating* if for all pairs $G_1 \not\simeq G_2 \in \mathcal{X}^{n\times n}$, there exists $h \in \mathcal{C}$ such that $h(G_1) \neq h(G_2)$.



In words, GIso-discriminating means being able to distinguish every pair of non-isomorphic graphs

Bridging the two perspectives

Universally approximating

A family $\mathcal{C}$ of invariant functions from $\mathcal{X}^{n\times n}$ to $\mathbb{R}$ is called *universally approximating* if for all f from $\mathcal{X}^{n\times n}$ to $\mathbb{R}$ which is invariant, and for all $\epsilon > 0$, there exists $h_{f,\epsilon} \in \mathcal{C}$ such that $\|f - h_{f,\epsilon}\|_\infty := \sup_{G\in\mathcal{X}^{n\times n}} |f(G) - h_{f,\epsilon}(G)| < \epsilon$

Now, we want to find connections between these two properties.

Augmenting a function family by appending NN layers

$\mathcal{C}^{+L}$

If $\mathcal{C}$ is a family of functions from $\mathcal{X}^{n \times n}$ to $\mathbb{R}$, consider the set of functions from graphs G to $\mathcal{NN}([h_1(G), ..., h_d(G)])$ for some finite d and $h_1, ..., h_d \in \mathcal{C}$, where $\mathcal{NN}$ is a feed-forward neural network with ReLU and L layers.

For example, if $\mathcal{C}_{L_0}$ is the collection of feed-forward neural networks with L_0 layers, then $\mathcal{C}_{L_0}^{+L}$ represents the collection of feed-forward neural networks with $L_0 + L$ layers.

GIso-discriminating equivalent to universally approximating

Theorem 1

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

Theorem 2

If $\mathcal{C}$ is GIso-discriminating, then $\mathcal{C}^{+2}$ is universally approximating.

For their proofs, please see section 3 and appendix A of the paper.

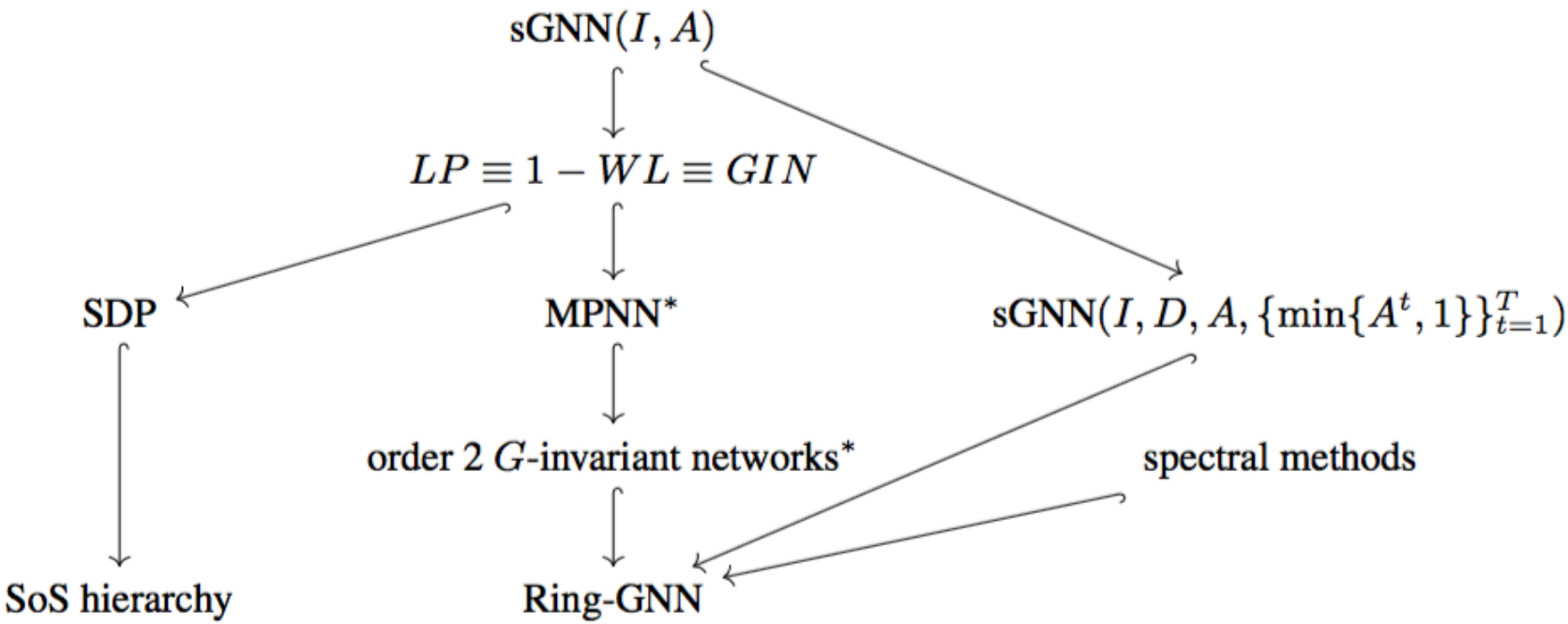
Sigma-algebra framework that unifies both perspectives

Furthermore, we propose a unifying framework for studying the expressive power of graph neural networks and other more general functions on graph data.

It is based on the concept of sigma-algebra, and the details are given in section 4 in the paper.

Comparing different families using this framework

This allows us to compare the representation power of different classes of GNNs succinctly, by comparing the σ-algebras generated by them.



Ring-GNN

In the rest of our work, we propose a new model, *Ring-GNN*, which is provably more powerful than another model, using the framework developed above.

Order-2 Graph G-invariant Networks

Definition

An *order-2 Graph G-invariant network* is a function $F : \mathbb{R}^{n\times n\times d_0} \to \mathbb{R}$ that can be decomposed in the following way:

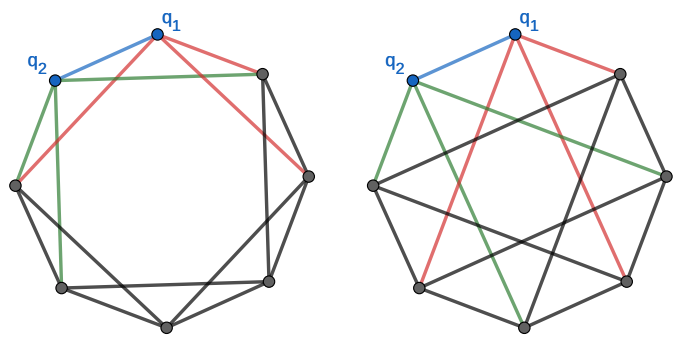
$$F = m \circ h \circ L_T \circ \sigma \circ \cdots \circ \sigma \circ L_1,$$

where each L_i is a linear graph G-equivariant layer from $\mathbb{R}^{n\times n\times d_{i-1}}$ to $\mathbb{R}^{n\times n\times d_i}$, σ is a pointwise activation function, h is a graph G-invariant layer from $\mathbb{R}^{n\times n\times d_T}$ to $\mathbb{R}$, and m is an MLP.

Limitations of order-2 Graph G-invariant Networks

Theorem 3

Order-2 Graph G-invariant Networks cannot distinguish between non-isomorphic regular graphs with the same degree.



For a detailed proof, please see appendix D.2.

Introducing Ring-GNN as an extension

Firstly, Maron et al. (2019) shows that all linear equivariant layers from $\mathbb{R}^{n\times n}$ to $\mathbb{R}^{n\times n}$ can be expressed as

$$L_\theta(A) = \sum_{i=1}^{15} \theta_i L_i(A) + \sum_{i=16}^{17} \theta_i \overline{L}_i,$$

where the $\{L_i\}_{i=1,\dots,15}$ are the 15 basis functions of all linear equivariant functions from $\mathbb{R}^{n\times n}$ to $\mathbb{R}^{n\times n}$, $\overline{L}_{16}$ and $\overline{L}_{17}$ are the basis for the bias terms, and $\theta \in \mathbb{R}^{17}$ are the parameters that determine L. Generalizing to an equivariant linear layer from $\mathbb{R}^{n\times n\times d}$ to $\mathbb{R}^{n\times n\times d'}$, we set

$$L_\theta(A)_{\cdot,\cdot,k'} = \sum_{k=1}^{d}\sum_{i=1}^{15} \theta_{k,k',i} L_i(A_{\cdot,\cdot,i}) + \sum_{i=16}^{17} \theta_{k,k',i} \overline{L}_i,$$

with $\theta \in \mathbb{R}^{d\times d'\times 17}$.

Introducing Ring-GNN as an extension

With this formulation, we now define a Ring-GNN with T layers. First, set $A^{(0)} = A$. In the t^{th} layer, let

$$\begin{aligned} B_1^{(t)} &= \sigma(L_{\alpha^{(t)}}(A^{(t)})) \\ B_2^{(t)} &= \sigma(L_{\beta^{(t)}}(A^{(t)}) \cdot L_{\gamma^{(t)}}(A^{(t)})) \\ A^{(t+1)} &= k_1^{(t)} B_1^{(t)} + k_2^{(t)} B_2^{(t)} \end{aligned}$$

where $k_1^{(t)}, k_2^{(t)} \in \mathbb{R}$, $\alpha^{(t)}, \beta^{(t)}, \gamma^{(t)} \in \mathbb{R}^{d^{(t)} \times d'^{(t)} \times 17}$ are learnable parameters, and σ is a pointwise nonlinearity such as ReLU.

If a scalar output is desired, then in the general form, we set the output to be $\theta_S \sum_{i,j} A_{ij}^{(T)} + \theta_D \sum_{i,i} A_{ii}^{(T)} + \sum_i \theta_i \lambda_i(A^{(T)})$, where $\theta_S, \theta_D, \theta_1, \ldots, \theta_n \in \mathbb{R}$ are trainable parameters, and $\lambda_i(A^{(T)})$ is the i-th eigenvalue of $A^{(L)}$. The third term, which is optional and task-dependent, can be computed via singular value decomposition.

Ring-GNN provably more powerful

We verify experimentally that Ring-GNN succeeds in distinguishing certain non-isomorphic regular graphs (the Circular Skip Link graphs), on which order-2 Graph G-invariant Networks fail, as we have shown.

Murphy, Srinivasan, Rao, Ribeiro (2019)

Experiments

GNN architecture	Circular Skip Links			IMDBB		IMDBM	
	max	min	std	mean	std	mean	std
RP-GIN †	53.3	10	12.9	-	-	-	-
GIN † ‡	10	10	0	75.1	5.1	52.3	2.8
Order-2 Graph G-inv. †	10	10	0	71.3	4.5	48.6	3.9
sGNN-5	80	80	0	72.8	3.8	49.4	3.2
sGNN-2	30	30	0	73.1	5.2	49.0	2.1
sGNN-1	10	10	0	72.7	4.9	49.0	2.1
LGNN	30	30	0	74.1	4.6	50.9	3.0
Ring-GNN	80	10	15.7	73.0	5.4	48.2	2.7
Ring-GNN (w/ degree) ‡	-	-	-	73.3	4.9	51.3	4.2

Experiments

	collab	mutag	ptc	proteins
Ring-GNN	80.1±1.4	86.8±6.4	65.7±7.1	75.7±2.9
GIN	80.2±1.9	89.4±5.6	64.6±7.0	76.2±2.8
Order-2 Graph G-inv.	77.9 ± 1.7	84.6±10.0	59.5±7.3	75.2±4.3

Thanks!

On the equivalence between graph isomorphism testing and function approximation with GNNs

Z. Chen, S. Villar, L. Chen, J. Bruna, NeurIPS 2019

`https://arxiv.org/abs/1905.12560`