How Powerful are Graph Neural Networks?

(EE531 Final Project - Graph)

K. Xu[1] W. Hu[2] J. Leskovec[2] S. Jegelka[1]

[1]MIT
[2]Stanford University

Appeared at ICLR 2019

Table of Contents

Outline

Introduction

- Since GNN(Graph Neural Network) has come out, it has revolutionized the field of representation learning, especially with graph datas.
- But why GNNs work? Why are they so powerful?
- Most interestingly, *how powerful are they?*

Outline

Previous / Related Works

- There hasn't been much work regarding this topic.
- Scarselli et al.[1] showed that the (probably) earliest GNN model[2] can approximate measurable functions in probability.
- Lei et al.[3] showed that their architecture lies in the RKHS of graph kernels, *but do not study explicitly which graph it can distinguish.*
- Above works focus on a specific architecture and **do not easily generalize to other architectures**.
- This paper presents a general framework for analyzing/characterizing the expressive power of a broad class of GNNs!

[1] Franco Scarselli et al. "The Graph Neural Network Model". In: *IEEE Trans. Neural Networks* 20.1 (2009), pp. 61–80.

[2] Franco Scarselli et al. "Computational Capabilities of Graph Neural Networks". In: *IEEE Trans. Neural Networks* 20.1 (2009), pp. 81–102.

[3] Tao Lei et al. "Deriving Neural Architectures from Sequence and Graph Kernels". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 2017, pp. 2024–2033.

Outline

Outline

GNN

- Let $G = (V, E)$ be a graph, or data with graph structure.
- Each $v \in V$ has a *node feature vector*, X_v
- There are two tasks of interest where GNN is commonly used:

Node Classification Problem

Each $v \in V$ has an associated label y_v.
Goal: Learn a representation vector h_v of v such that $y_v = f(h_v)$ i.e. such that v's label can be predicted.

Graph Classification Problem

A set of graphs $\{G_1, \ldots, G_N\} \subset \mathcal{G}$ is given, along with their labels $\{y_1, \ldots, y_N\} \subset \mathcal{Y}$.
Goal: Learn a representation vector h_G of G such that $y_G = f(h_G)$ i.e. such that G's label can be predicted.

GNN

- Modern GNNs follow a neighborhood aggregation strategy (message passing strategy)
- Iteratively update the representation of a nodes by aggregating representations of its neighbors!
- Let $h_v^{(k)}$ is the feature vector of node v at the k-th iteration/layer, and let us initialize it as $h_v^{(0)} = X_v$.
- k-th layer of a GNN is

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

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

GNN

- Different choices of AGGREGATE$^{(k)}$ and COMBINE$^{(k)}$ have led to different GNN variants/architectures.
- GraphSAGE[4]:

$$a_v^{(k)} = \mathrm{MAX}\left(\left\{\mathrm{ReLU}\left(W h_u^{(k-1)}\right) : u \in \mathcal{N}_G(v)\right\}\right)$$

$$h_v^{(k)} = W\left[h_v^{(k-1)}, a_v^{(k)}\right]$$

- Graph Convolutional Networks, or GCN[5]:

$$h_v^{(k)} = \mathrm{ReLU}\left(W\,\mathrm{MEAN}\left\{h_u^{(k-1)} : u \in \mathcal{N}_G(v) \cup \{v\}\right\}\right)$$

---

[4] William L. Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pp. 1024–1034.

[5] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017.

GNN

- In case of node classification, the final node representation $h_v^{(K)}$ is used for prediction.
- In case of graph classification, the final node representations are aggregated by READOUT function to obtain the entire graph's representation:

$$h_G = \mathrm{READOUT}\left(\left\{h_v^{(K)} : v \in V\right\}\right)$$

- READOUT can be a simple permutation invariant function, or something more sophisticated[6][7]

[6] Zhitao Ying et al. "Hierarchical Graph Representation Learning with Differentiable Pooling". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*. 2018, pp. 4805–4815.

[7] Muhan Zhang et al. "An End-to-End Deep Learning Architecture for Graph Classification". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. 2018, pp. 4438–4445.

Outline

Graph Isomorphism Problem

- Consider the following problem:

GRAPH ISOMORPHISM (GI)

Input: Two finite graphs G_1 and G_2
Question: $G_1 \cong G_2$?

- Appears in: discrete mathematics, mathematical logic, theory of computation, machine learning, computer vision...etc.
- This seemingly harmless problem has harassed researchers for decades!

Graph Isomorphism Problem

Here are some facts related to GI:

- Not known to be of class NP-complete nor tractable!
  (Researchers have actually defined a new complexity class **GI**)
- It is currently known that GI can be solved in quasipolynomial time *i.e.* in $O\left(2^{O((\log n)^c)}\right)$ $(c > 0)$ time[8]:

Theorem (Babai, 2015)

The Graph Isomorphism problem ... can be solved in quasipolynomial time.

(Confirmed by Harald Andrés Helfgott, probably correct)

- But it is not practical!
- Some practical algorithms: McKay (1981), Schmidt & Druffel (1976), Ullman (1976)...etc.

---

[8] László Babai. "Graph isomorphism in quasipolynomial time [extended abstract]". In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. 2016, pp. 684–697.

Weisfeiler-Lehman test

- Weisfeiler-Lehman test of graph isomorphism[9], or simply WL test, is a combinatorial algorithm for GI.
- WL test is proved to be successful (and computationally efficient) in isomorphism testing for a broad class of graphs[10]
- There are some cases (ex. regular graphs) when the WL test fails[11]

[9] Boris Weisfeiler and Andrei A. Lehman. "A reduction of a graph to a canonical form and an algebra arising during this reduction". In: *Nauchno-Technicheskaya Informatsia* 2.9 (1968), pp. 12–16.

[10] László Babai and Ludek Kucera. "Canonical Labelling of Graphs in Linear Average Time". In: *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*. 1979, pp. 39–46.

[11] Jin-yi Cai, Martin Fürer, and Neil Immerman. "An optimal lower bound on the number of variables for graph identifications". In: *Combinatorica* 12.4 (1992), pp. 389–410.

Weisfeiler-Lehman test

- Why are we interested in this WL test?
- 1-dimensional form of the WL test ("naïve vertex refinement") is *based on neighbor aggregations*, analogous to the GNNs!
- Overview of the algorithm:
  - Aggregate the labels of nodes and their neighborhoods
  - Hashes the aggregated label into *unique* new labels
  - If at some iteration the labels of the nodes between the two graphs differ, then the two graphs are non-isomorphic.

1-dim WL test

Let (G, l) be a labeled graph i.e. a graph G with an endowed node coloring $l : V(G) \to \Sigma$. (Σ: arbitrary codomain)

- At t-th iteration ($t \geq 0$), the 1-WL computes a node coloring $c_l^{(t)} : V(G) \to \Sigma$, which depends on the previous node coloring:

  $$c_l^{(0)} = l, \ c_l^{(t)}(v) = \text{HASH}\left(\left(c_l^{(t-1)}(v), \{\{c_l^{(t-1)}(u) | u \in \mathcal{N}(v)\}\}\right)\right)$$

  (HASH bijectively maps the above pair to a unique value in Σ that hasn't been used in previous iterations)
- Run above algorithm in parallel for the two input graphs.
- If at some iteration, the two graphs have a different number of nodes colored $\sigma \in \Sigma$, conclude that the graphs are not isomorphic.
  (This why this 1-dim version is commonly called the *color refinement algorithm*)

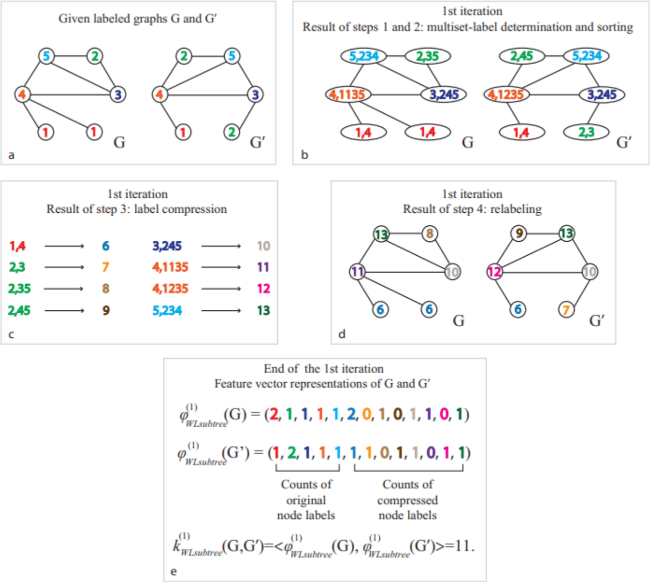
WL subtree kernel

- Graph kernel: kernel function that defines *inner product on graphs*[12] (Function measuring the similarity of a pair of two given graphs)
- Some examples: *random walk* (Gärtner *et al.*, 2003; Borgwardt *et al.*, 2005), *marginalized* (Kashima *et al.*, 2003, 2004; Mahé *et al.*, 2004)
- Weisfeiler-Lehman subtree kernel[13]: counts common *original and compressed labels* (resulting from 1-dim WL test) in two graphs.

[12] S. V. N. Vishwanathan et al. "Graph Kernels". In: *J. Mach. Learn. Res.* 11 (2010), pp. 1201–1242.

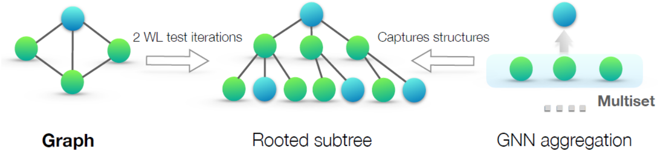
[13] Nino Shervashidze et al. "Weisfeiler-Lehman Graph Kernels". In: *J. Mach. Learn. Res.* 12 (2011), pp. 2539–2561.

WL subtree kernel



WL subtree kernel

- Is it related to GNN? Yes!
- The kernel uses the *counts of node labels* at different iterations of the WL test as the *feature vector* of a graph.
- Intuitively, a node's label at the k-th iteration of the 1-dim WL test represents a subtree structure of height k rooted at the node.



- Thus, the graph features considered by the WL subtree kernel are essentially counts of different rooted subtrees in the graph!

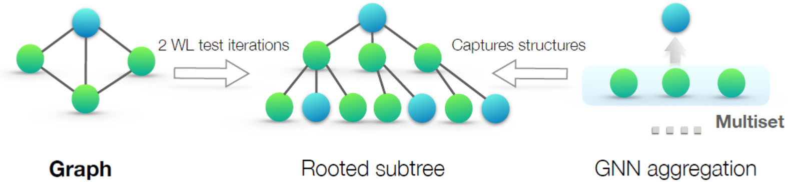
k-dim WL test

- k-dim WL test is a generalization of the 1-dim WL test; it colors tuples from $V(G)^k$ instead of nodes.
- Why would we want to do that?
- By increasing k, the algorithm gets more powerful in terms of distinguishing non-isomorphic graphs!
- It was shown that for each $k \geq 2$, there are non-isomorphic graphs which can be distinguished by the $(k+1)$-dim WL test, but not by the k-dim WL test[14]
- In this work, we only focus on 1-dim WL test.

[14] Jin-yi Cai, Martin Fürer, and Neil Immerman. "An optimal lower bound on the number of variables for graph identifications". In: *Combinatorica* 12.4 (1992), pp. 389–410.

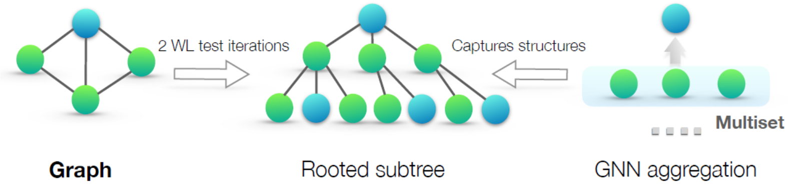
Outline

(Overview of) Theoretical Framework



- GNN: recursive update of each node's feature vector *i.e.* its rooted subtree structure!
- (1-dim) WL test: also results in rooted subtree structure!
- Assign each feature vector a unique label from a countable universe.
- Then, feature vectors of a set of neighboring nodes form a multiset.

(Overview of) Theoretical Framework



- Representational power of a GNN: when a GNN maps two nodes to the same location (in the embedding space)?
- Maximally powerful GNN: its aggregation scheme must be injective!
- Closely related to GRAPH ISOMORPHISM.
- GNN's aggregation scheme: *class of functions over multisets that their neural networks can represent*

Outline

Representational capacity of GNNs

- Recall: maximally powerful GNN has injective aggregation scheme.
- Two (non)isomorphic graphs are mapped to the (different)same representation(s).
- Characterized by GRAPH ISOMORPHISM!

Lemma 2

Let G_1 and G_2 be any two non-isomorphic graphs.
If a graph neural network $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ maps G_1 and G_2 to different embeddings, the Weisfeiler-Lehman graph isomorphism test also decides G_1 and G_2 are not isomorphic.

- It says that any aggregation-based GNN is *at most* as powerful as the WL test in distinguishing different graphs.

Representational capacity of GNNs

- Is that "bound" tight?
- In other words, does there exist GNN that is, in principle, as powerful as the WL test in distinguishing different graphs?

Theorem 3

Let $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ be a GNN.
With a sufficient number of GNN layers, $\mathcal{A}$ maps any G_1 and G_2 that the Weisfeiler-Lehman test of isomorphism decides as non-isomorphic, to different embeddings if the following conditions hold:

- $\mathcal{A}$ aggregates and updates node features iteratively with

$$h_v^{(k)} = \phi\left(h_v^{(k-1)}, f\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}_G(v)\right\}\right)\right)$$

  where the functions f, which operates on multisets, and ϕ are *injective*.

- $\mathcal{A}$'s graph-level readout, which operates on the multiset of node features $\left\{h_v^{(k)}\right\}$, is *injective*.

Representational capacity of GNNs

- Node feature vectors in the WL test are essentially one-hot encodings, and thus cannot capture similarity between subtrees!
- GNN (satisfying the criteria in Theorem 3) generalizes the WL test by *learning to embed* the subtrees to low-dimensional space.
- GNNs can not only discriminate different structures, but can also learn to map similar graph structures to similar embeddings and *capture dependencies between graph structures*.
- Especially useful when co-occurence of subtrees is sparse across different graphs, or there are noisy edges and node features.[15]

[15] Pinar Yanardag and S. V. N. Vishwanathan. "Deep Graph Kernels". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. 2015, pp. 1365–1374.

Outline

Graph Isomorphism Network (GIN)

- Well we've proved (or more accurately, seen) that GNNs under certain conditions is maximally powerful.
- Let us develop a simple architecture, GIN!
- Idea: deep multisets[16] *i.e.* parametrizing universal multiset functions with neural networks.

[16] Manzil Zaheer et al. "Deep Sets". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pp. 3391–3401.

Graph Isomorphism Network (GIN)

Lemma 5

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 $g(X) = \phi\left(\sum_{x \in X} f(x)\right)$ for some function ϕ.

- Observe that certain popular injective set functions, such as the mean aggregator, are *not* injective multiset functions!
- This lemma tells us that sum aggregators can represent injective, in fact, *universal* functions over multisets.
- Thus, we can conceive aggregation schemes that can represent universal functions over a node and the multiset of its neighbors, satisfying the *injectiveness condition* (a) in Theorem 3!

Graph Isomorphism Network (GIN)

- Here is a simple and concrete formulation of the previous discussion:

Corollary 6

Assume $\mathcal{X}$ is countable. There exists a function $f : \mathcal{X} \to \mathbb{R}^n$ so that for infinitely many choices of ϵ, including all irrational numbers, $h(c, X) = (1 + \epsilon)f(c) + \sum_{x \in X} f(x)$ is unique for each pair (c, X), where $c \in \mathcal{X}$ and $X \subset \mathcal{X}$ is a multiset of bounded size.
Moreover, any function g over such pairs can be decomposed as $g(c, X) = \varphi\left((1 + \epsilon)f(c) + \sum_{x \in X} f(x)\right)$ for some function φ.

- We can use MLPs to model and learn f and φ, thanks to the Universal Approximation Theorem[17][18].

[17] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366.

[18] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251–257.

How powerful is MLP?

Universal Approximation Theorem (Hornik, 1991)

Define

$$\mathscr{N}_k^{(n)}(\psi) = \left\{ h : \mathbb{R}^k \to \mathbb{R} \,\middle|\, h(x) = \sum_{j=1}^{n} \beta_j \psi(a_j' x - \theta_j) \right\}$$

as the set of all functions implemented by such a network with n hidden units, where ψ is the common activation function of the hidden units.
If ψ is continuous, bounded and nonconstant, then $\mathscr{N}_k^{(n)}(\psi)$ is dense in $\mathscr{C}(X)$ for all compact subsets X of $\mathbb{R}^k$.

- Every continuous function can be approximated arbitrarily closely by a multi-layer perceptron with just one hidden layer.
- The choice of the activation function doesn't matter; it's the multilayer feedforward architecture that gives neural networks the potential of being universal approximators.

Graph Isomorphism Network (GIN)

- In practice, $f^{(k+1)} \circ \varphi^{(k)}$ is modeled with one MLP.
- We may make ϵ as a learnable parameter, or a fixed scalar.
- Then, GIN updates node representations as:

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

Graph Isomorphism Network (GIN)

- Node embeddings, learned by the GIN, can be directly used for node classification and link prediction.
- For graph classification tasks, we need a READOUT function.
- We want to consider all structural information, considering that features from earlier iterations may sometimes generalize better.
- Use information from *all* depths/iterations of the model[19]!

$$h_G = \text{CONCAT}\left(\text{READOUT}\left(\left\{h_v^k | v \in V(G)\right\}\right)\middle| k = 0, 1, \ldots, K\right)$$

- Note that if GIN replaces READOUT with summing all node features from the same iteration, it provably generalizes the WL test and the WL subtree kernel.

[19] Keyulu Xu et al. "Representation Learning on Graphs with Jumping Knowledge Networks". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. 2018, pp. 5449–5458.

Outline

Overview

- Now we consider GNNs that do not satisfy the conditions as described in Theorem 3 and/or GNNs with different choice of AGGREGATE (Max-pooling, Mean)
- What if 1-layer perceptron is used instead of MLPs?
- What if the sum $h(X) = \sum_{x \in X} f(x)$ is replaced by mean/max pooling?

Outline

MLP?

- The function f in Lemma 5 helps map distinct multisets to unique embeddings.
- f can be parametrized by MLPs, as shown by the Universal Approximation Theorem[20]

[20] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251–257.

Without MLP?

- Many modern GNNs, however, use a *1-layer perceptron* $\sigma \circ W$: a linear mapping followed by a non-linear activation function.
- Is 1-layer perceptron enough for graph learning?

Lemma 7

There exist finite multisets $X_1 \neq X_2$ so that for any linear mapping W, $\sum_{x \in X_1} \mathrm{ReLU}(Wx) = \sum_{x \in X_2} \mathrm{ReLU}(Wx)$

- Unlike models using MLPs, 1-layer perceptron (even with the bias term) is not a universal approximator of multiset functions.

Outline

GCN

- As described previously, Graph Convolutional Network[21] takes the form:

$$h_v^{(k)} = \mathrm{ReLU}\left(W\,\mathrm{MEAN}\left\{h_u^{(k-1)} : u \in \mathcal{N}_G(v) \cup \{v\}\right\}\right)$$

- GCN utilizes mean aggregator.
- How can we characterize the structures that GCN can or cannot capture?

[21] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017.

Mean aggregator

- Consider two multisets $X_1 = (S, m)$ and $X_2 = (S, km)$
- Observation: Any mean aggregator maps X_1 and X_2 to the *same* embeddings!
- Mean aggregator captures the distribution of elements in a multiset.

Corollary 8

Assume $\mathcal{X}$ is countable. There exists a function $f : \mathcal{X} \to \mathbb{R}^n$ so that $h(X) = \frac{1}{|X|} \sum_{x \in X} f(x)$, $h(X_1) = h(X_2)$ *if only if* multisets X_1 and X_2 hav e the same distribution. That is, assuming $|X_2| \geq |X_1|$, we have $X_1 = (S, m)$ and $X_2 = (S, km)$ for some $k \in \mathbb{N}$.

- This is as powerful as the sum aggregator if the node features are diverse and rarely repeat, and thus *effective for node classification*.

Outline

GraphSAGE

- As described previously, GraphSAGE[22] takes the form:

$$a_v^{(k)} = \mathrm{MAX}\left(\left\{\mathrm{ReLU}\left(Wh_u^{(k-1)}\right) : u \in \mathcal{N}_G(v)\right\}\right)$$

$$h_v^{(k)} = W\left[h_v^{(k-1)}, a_v^{(k)}\right]$$

- GraphSAGE utilizes max-pooling aggregator.
- How can we characterize the structures that GraphSAGE can or cannot capture?

[22] William L. Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pp. 1024–1034.

Max-pooling aggregator

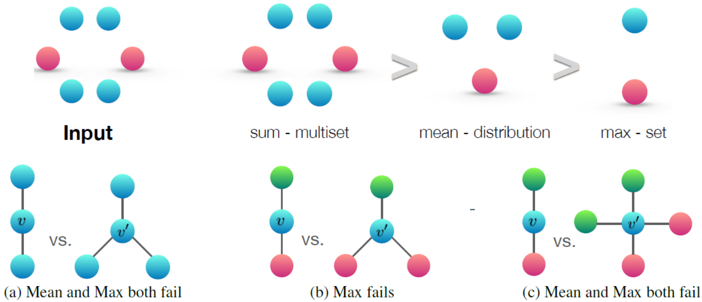
- Unlike previous aggregators, max-pooling can't capture exact structure nor the distribution!
- But it can capture the underlying set of muliset *i.e.* S in $X = (S, m)$

Corollary 9

Assume $\mathcal{X}$ is countable. There exists a function $f : \mathcal{X} \to \mathbb{R}^\infty$ so that $h(X) = \max_{x \in X} f(x)$, $h(X_1) = h(X_2)$ *if only if* multisets X_1 and X_2 have the same underlying set.

Summary

- Let us rank the three aggregators by their representational power:



- Sum over multiset aggregator (as in GIN) completely captures the exact structure of graph.
- Mean aggregator (as in GCN) captures the statistical and distributional information of the graph.
- Max-pooling aggregator (as in GraphSAGE) captures the representative elements of the graph, or its skeleton

Outline

Experiments

The goal of the experiment is to compare the training and test performance of GIN and less powerful GNN variants.

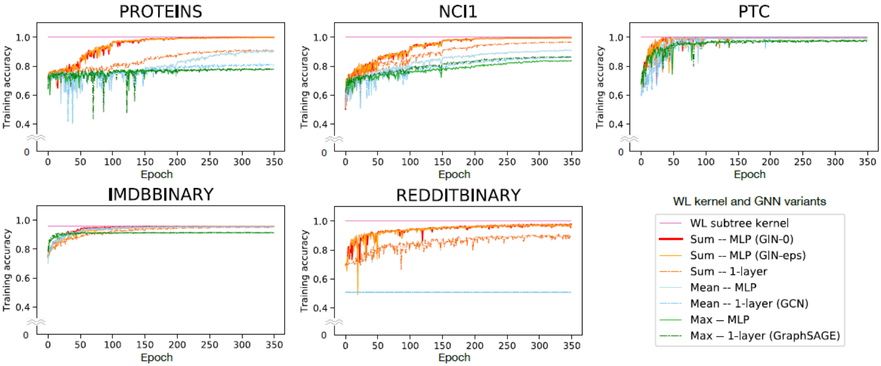
- Training set performance: compare different GNN models based on their *representational power*
- Test set performance: quantifies *generalization ability*

Experiment Design

- 9 graph classification benchmarks were used[23]:
  - 4 bioinformatics datasets (*MUTAG, PTC, NCI1, PROTEINS*)
  - 5 social network datasets (*COLLAB, IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, REDDIT-MULTI5K*)
- Several models were used:
  - GIN $-\epsilon$: GIN that *learns* ϵ by gradient descent
  - GIN -0: GIN that fixes ϵ to 0.
  - Architectures that replace the sum in the GIN -0 aggregation with mean or max-pooling, or replace MLPs with 1-layer perceptrons
  - GCN
  - GraphSAGE
- The baselines used were:
  - WL subtree kernel with C-SVM used as a classifier
  - Deep learning architectures i.e. DCNN, PATCHY-SAN, DGCNN
  - AWL

[23] Pinar Yanardag and S. V. N. Vishwanathan. "Deep Graph Kernels". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. 2015, pp. 1365–1374.

Results


PROTEINS
NCI1
PTC
IMDBBINARY
REDDITBINARY
Training accuracy
Epoch
WL kernel and GNN variants
WL subtree kernel
Sum -- MLP (GIN-0)
Sum -- MLP (GIN-eps)
Sum -- 1-layer
Mean -- MLP
Mean -- 1-layer (GCN)
Max -- MLP
Max -- 1-layer (GraphSAGE)

Results

	Datasets	IMDB-B	IMDB-M	RDT-B	RDT-M5K	COLLAB	MUTAG	PROTEINS	PTC	NCI1
Datasets	# graphs	1000	1500	2000	5000	5000	188	1113	344	4110
	# classes	2	3	2	5	3	2	2	2	2
	Avg # nodes	19.8	13.0	429.6	508.5	74.5	17.9	39.1	25.5	29.8
Baselines	WL subtree	73.8 ± 3.9	50.9 ± 3.8	81.0 ± 3.1	52.5 ± 2.1	78.9 ± 1.9	90.4 ± 5.7	75.0 ± 3.1	59.9 ± 4.3	**86.0 ± 1.8** *
	DCNN	49.1	33.5	–	–	52.1	67.0	61.3	56.6	62.6
	PatchySan	71.0 ± 2.2	45.2 ± 2.8	86.3 ± 1.6	49.1 ± 0.7	72.6 ± 2.2	**92.6 ± 4.2** *	75.9 ± 2.8	60.0 ± 4.8	78.6 ± 1.9
	DGCNN	70.0	47.8	–	–	73.7	85.8	75.5	58.6	74.4
	AWL	74.5 ± 5.9	51.5 ± 3.6	87.9 ± 2.5	54.7 ± 2.9	73.9 ± 1.9	87.9 ± 9.8	–	–	–
GNN variants	Sum–MLP (**GIN-0**)	**75.1 ± 5.1**	**52.3 ± 2.8**	**92.4 ± 2.5**	**57.5 ± 1.5**	**80.2 ± 1.9**	**89.4 ± 5.6**	**76.2 ± 2.8**	**64.6 ± 7.0**	**82.7 ± 1.7**
	Sum–MLP (**GIN-ϵ**)	**74.3 ± 5.1**	**52.1 ± 3.6**	**92.2 ± 2.3**	**57.0 ± 1.7**	**80.1 ± 1.9**	**89.0 ± 6.0**	**75.9 ± 3.8**	63.7 ± 8.2	**82.7 ± 1.6**
	Sum–1-Layer	74.1 ± 5.0	**52.2 ± 2.4**	90.0 ± 2.7	55.1 ± 1.6	**80.6 ± 1.9**	**90.0 ± 8.8**	**76.2 ± 2.6**	63.1 ± 5.7	82.0 ± 1.5
	Mean–MLP	73.7 ± 3.7	**52.3 ± 3.1**	50.0 ± 0.0	20.0 ± 0.0	79.2 ± 2.3	83.5 ± 6.3	75.5 ± 3.4	**66.6 ± 6.9**	80.9 ± 1.8
	Mean–1-Layer (GCN)	74.0 ± 3.4	51.9 ± 3.8	50.0 ± 0.0	20.0 ± 0.0	79.0 ± 1.8	85.6 ± 5.8	76.0 ± 3.2	64.2 ± 4.3	80.2 ± 2.0
	Max–MLP	73.2 ± 5.8	51.1 ± 3.6	–	–	–	84.0 ± 6.1	76.0 ± 3.2	64.6 ± 10.2	77.8 ± 1.3
	Max–1-Layer (GraphSAGE)	72.3 ± 5.3	50.9 ± 2.2	–	–	–	85.1 ± 7.6	75.9 ± 3.2	63.9 ± 7.7	77.7 ± 1.5

Outline

Summary

In summary,

- Theoretical foundations for reasoning about the expressive power of GNNS
- Tight bounds on the representational capacity of popular GNN variants. (cf. WL test)
- Designed a provably maximally powerful GNN under the neighborhood aggregation framework (*Graph Isomorphism Network*)

Future research

- Different aggregators?
- Go beyond neighborhood aggregation
- Understand/improve the generalization properties of GNNs
- What if the node features are continuous (uncountable)?
- Better understanding of the optimization landscape

Outline

Summary of GNNs

Here is a summary of "major" GNNs[24]: (next page)

[24] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *arXiv e-prints* (Jan. 2019). arXiv: 1901.00596 [cs.LG].

Summary of GNNs

(1: RecGNN, 2: Spectral-based ConvGNN, 3: Spatial-based ConvGNN)

Method	Category	Time Complexity	Features
GNN[16]	1	$O(m)$	Information diffusion mechanism, updates nodes' states until a stable equilibrium is reached.
Spectral CNN[4]	2	$O(n^3)$	Treats the filters as a set of learnable parameters.
ChebNet[6]	2	$O(m)$	Approximates the filger by Chebyshev polynomials of the diagonal matrix of eigenvalues.
GCN[11]	2	$O(m)$	First-order approximation of ChebNet
AGCN[13]	2	$O(n^2)$	Learns hidden structural relations by using the residual graph adjacency matrix through learnable metric.
DualGCN[27]	2	$O(m)$	Introduces dual graph convolutional architecture with two graph convolutional layers in parallel.
NN4G[14]	3	$O(m)$	Performs graph convolutions by summing up a node's neighborhood information directly.
DCNN[1]	3	$O(n^2)$	Treats graph convolutions as a diffusion process
MPNN[7]	3	$O(m)$	Treats graph convolutions as a message passing process.
GraphSAGE[8]	3	-	Sampling of fixed number of neighbors for each node.
GIN[21]	3	$O(m)$	This paper!

Outline

References I

[1] James Atwood and Don Towsley. "Diffusion-Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016, pp. 1993–2001.

[2] László Babai. "Graph isomorphism in quasipolynomial time [extended abstract]". In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. 2016, pp. 684–697.

[3] László Babai and Ludek Kucera. "Canonical Labelling of Graphs in Linear Average Time". In: *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*. 1979, pp. 39–46.

[4] Joan Bruna et al. "Spectral Networks and Locally Connected Networks on Graphs". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014.

[5] Jin-yi Cai, Martin Fürer, and Neil Immerman. "An optimal lower bound on the number of variables for graph identifications". In: *Combinatorica* 12.4 (1992), pp. 389–410.

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2016, pp. 3837–3845.

[7] Justin Gilmer et al. "Neural Message Passing for Quantum Chemistry". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 2017, pp. 1263–1272.

[8] William L. Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pp. 1024–1034.

[9] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251–257.

[10] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366.

References II

[11] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017.

[12] Tao Lei et al. “Deriving Neural Architectures from Sequence and Graph Kernels”. In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 2017, pp. 2024–2033.

[13] Ruoyu Li et al. “Adaptive Graph Convolutional Neural Networks”. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. 2018, pp. 3546–3553.

[14] Alessio Micheli. “Neural Network for Graphs: A Contextual Constructive Approach”. In: *IEEE Trans. Neural Networks* 20.3 (2009), pp. 498–511.

[15] Franco Scarselli et al. “Computational Capabilities of Graph Neural Networks”. In: *IEEE Trans. Neural Networks* 20.1 (2009), pp. 81–102.

[16] Franco Scarselli et al. “The Graph Neural Network Model”. In: *IEEE Trans. Neural Networks* 20.1 (2009), pp. 61–80.

[17] Nino Shervashidze et al. “Weisfeiler-Lehman Graph Kernels”. In: *J. Mach. Learn. Res.* 12 (2011), pp. 2539–2561.

[18] S. V. N. Vishwanathan et al. “Graph Kernels”. In: *J. Mach. Learn. Res.* 11 (2010), pp. 1201–1242.

[19] Boris Weisfeiler and Andrei A. Lehman. “A reduction of a graph to a canonical form and an algebra arising during this reduction”. In: *Nauchno-Technicheskaya Informatsia* 2.9 (1968), pp. 12–16.

[20] Zonghan Wu et al. “A Comprehensive Survey on Graph Neural Networks”. In: *arXiv e-prints* (Jan. 2019). arXiv: `1901.00596 [cs.LG]`.

[21] Keyulu Xu et al. “How Powerful are Graph Neural Networks?” In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. 2019.

[22] Keyulu Xu et al. "Representation Learning on Graphs with Jumping Knowledge Networks". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. 2018, pp. 5449–5458.

[23] Pinar Yanardag and S. V. N. Vishwanathan. "Deep Graph Kernels". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. 2015, pp. 1365–1374.

[24] Zhitao Ying et al. "Hierarchical Graph Representation Learning with Differentiable Pooling". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*. 2018, pp. 4805–4815.

[25] Manzil Zaheer et al. "Deep Sets". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pp. 3391–3401.

[26] Muhan Zhang et al. "An End-to-End Deep Learning Architecture for Graph Classification". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. 2018, pp. 4438–4445.

[27] Chenyi Zhuang and Qiang Ma. "Dual Graph Convolutional Networks for Graph-Based Semi-Supervised Classification". In: *Proceedings of the 2018 World Wide Web Conference*. WWW '18. Lyon, France: International World Wide Web Conferences Steering Committee, 2018, pp. 499–508. ISBN: 978-1-4503-5639-8.

Thank you for your attention! Any questions?