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

1. Introduction
2. Preliminaries
   - GNNs
   - WL test
   - (Overview of) Theoretical Framework
3. Representational capacity of GNNs
4. Graph Isomorphism Network (GIN)
5. Less powerful, but still interesting GNNs
   - 1-Layer Perceptrons
   - GCN
   - GraphSAGE
6. Experiments
7. Summary
8. Future Research
9. Roles and Responsibilities

Outline

1. Introduction
2. Preliminaries
   - GNNs
   - WL test
   - (Overview of) Theoretical Framework
3. Representational capacity of GNNs
4. Graph Isomorphism Network (GIN)
5. Less powerful, but still interesting GNNs
   - 1-Layer Perceptrons
   - GCN
   - GraphSAGE
6. Experiments
7. Summary
8. Future Research
9. Roles and Responsibilities

Introduction

- Since GNN(Graph Neural Network) has come out, it has revolutionized the field of representation learning, especially with graph datas.

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?

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

1. Introduction
2. Preliminaries
   - GNNs
   - WL test
   - (Overview of) Theoretical Framework
3. Representational capacity of GNNs
4. Graph Isomorphism Network (GIN)
5. Less powerful, but still interesting GNNs
   - 1-Layer Perceptrons
   - GCN
   - GraphSAGE
6. Experiments
7. Summary
8. Future Research
9. Roles and Responsibilities

Outline

1. Introduction
2. Preliminaries
   - GNNs
   - WL test
   - (Overview of) Theoretical Framework
3. Representational capacity of GNNs
4. Graph Isomorphism Network (GIN)
5. Less powerful, but still interesting GNNs
   - 1-Layer Perceptrons
   - GCN
   - GraphSAGE
6. Experiments
7. Summary
8. Future Research
9. Roles and Responsibilities

GNN

- Let $G = (V, E)$ be a graph, or data with graph structure.

GNN

- Let $G = (V, E)$ be a graph, or data with graph structure.
- Each $v \in V$ has a *node feature vector*, X_v

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:

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.

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)

GNN

- Modern GNNs follow a neighborhood aggregation strategy (message passing strategy)
- Iteratively update the representation of a nodes by aggregating representations of its neighbors!

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 $h_v^{(0)} = X_v$.

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 $\text{AGGREGATE}^{(k)}$ and $\text{COMBINE}^{(k)}$ have led to different GNN variants/architectures.

GNN

- Different choices of AGGREGATE$^{(k)}$ and COMBINE$^{(k)}$ have led to different GNN variants/architectures.
- GraphSAGE (Hamilton *et al.*, 2017):

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

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

GNN

- Different choices of AGGREGATE$^{(k)}$ and COMBINE$^{(k)}$ have led to different GNN variants/architectures.
- GraphSAGE (Hamilton *et al.*, 2017):

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

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

- Graph Convolutional Networks, or GCN (Kipf & Welling, 2017):

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

GNN

- In case of node classification, the final node representation $h_v^{(K)}$ is used for prediction.

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:

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

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:

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

- READOUT can be a simple permutation invariant function, or something more sophisticated (cf. Ying *et al.*, 2018; Zhang *et al.*, 2018)

Outline

1. Introduction
2. Preliminaries
   - GNNs
   - WL test
   - (Overview of) Theoretical Framework
3. Representational capacity of GNNs
4. Graph Isomorphism Network (GIN)
5. Less powerful, but still interesting GNNs
   - 1-Layer Perceptrons
   - GCN
   - GraphSAGE
6. Experiments
7. Summary
8. Future Research
9. Roles and Responsibilities

Graph Isomorphism Problem

- Consider the following problem:

Graph Isomorphism Problem

- Consider the following problem:

GRAPH ISOMORPHISM (GI)

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

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.

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**)

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:

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:

Theorem (Babai, 2015)

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

(Confirmed by Harald Andrés Helfgott, probably correct)

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:

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!

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:

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.

Weisfeiler-Lehman test

- Weisfeiler-Lehman test of graph isomorphism (Weisfeiler & Lehman, 1968), or simply WL test, is a combinatorial algorithm for GI.

Weisfeiler-Lehman test

- Weisfeiler-Lehman test of graph isomorphism (Weisfeiler & Lehman, 1968), 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 (Babai & Kucera, 1979)

Weisfeiler-Lehman test

- Weisfeiler-Lehman test of graph isomorphism (Weisfeiler & Lehman, 1968), 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 (Babai & Kucera, 1979)
- There are some cases (ex. regular graphs) when the WL test fails (Cai *et at.*, 1992)

Weisfeiler-Lehman test

- Why are we interested in this WL test?

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!

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:

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

(HASH bijectively maps the above pair to a unique value in Σ that hasn't been used in previous iterations)

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:

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

  (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.

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:

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

  (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.

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

- A graph kernel is a kernel function that computes an *inner product on graphs* (Vishwanathan *et al.*, 2010)
  (Functions measuring the similarity of pairs of graphs)

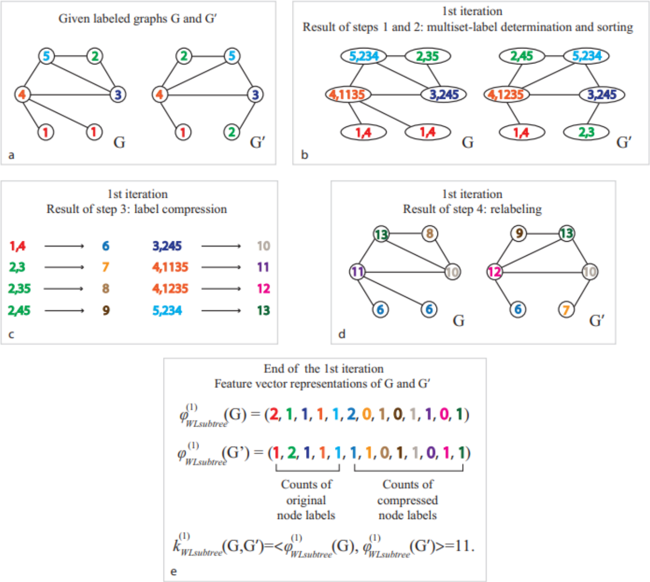
WL subtree kernel

- A graph kernel is a kernel function that computes an *inner product on graphs* (Vishwanathan *et al.*, 2010)
  (Functions measuring the similarity of pairs of graphs)
- Ex) random walk (Gärtner *et al.*, 2003; Borgwardt *et al.*, 2005), marginalized (Kashima *et al.*, 2003, 2004; Mahé *et al.*, 2004)

WL subtree kernel

- A graph kernel is a kernel function that computes an *inner product on graphs* (Vishwanathan *et al.*, 2010)
  (Functions measuring the similarity of pairs of graphs)
- Ex) random walk (Gärtner *et al.*, 2003; Borgwardt *et al.*, 2005), marginalized (Kashima *et al.*, 2003, 2004; Mahé *et al.*, 2004)
- Weisfeiler-Lehman subtree kernel (Shervashidze *et al.*, 2011): counts common *original and compressed labels* (resulting from 1-dim WL test) in two graphs.

WL subtree kernel



WL subtree kernel

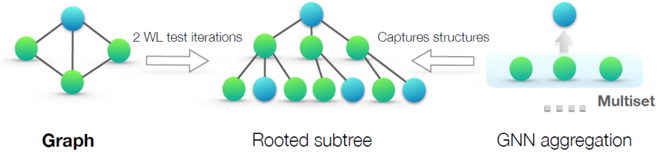
- Is it related to GNN? Yes!

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.

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.

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 be want to do that?

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 be want to do that?
- By increasing k, the algorithm gets more powerful in terms of distinguishing non-isomorphic graphs!

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 be 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 (Cai *et al.*, 1992)

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 be 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 (Cai *et al.*, 1992)
- In this work, we only focus on 1-dim WL test.

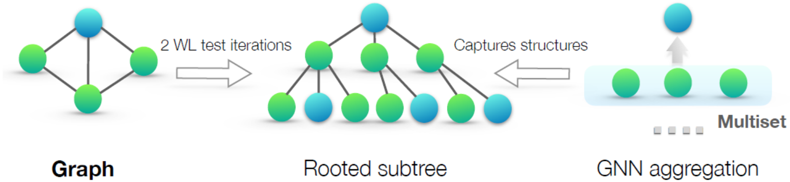
Outline

1 Introduction
2 Preliminaries
- GNNs
- WL test
- (Overview of) Theoretical Framework

3 Representational capacity of GNNs
4 Graph Isomorphism Network (GIN)
5 Less powerful, but still interesting GNNs
- 1-Layer Perceptrons
- GCN
- GraphSAGE

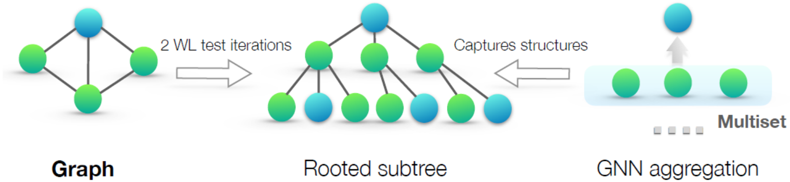
6 Experiments
7 Summary
8 Future Research
9 Roles and Responsibilities

(Overview of) Theoretical Framework



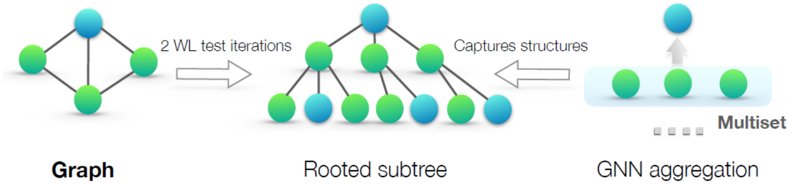
- GNN: recursive update of each node's feature vector *i.e.* its rooted subtree structure!

(Overview of) Theoretical Framework



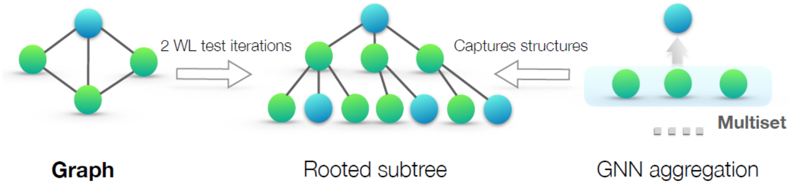
- GNN: recursive update of each node's feature vector *i.e.* its rooted subtree structure!
- Wl test: also results in rooted subtree! structure!
- Assign each feature vector a unique label from a countable universe.
- Then, feature vectors of a set of neighboring nodes form a *multiset*

(Overview of) Theoretical Framework



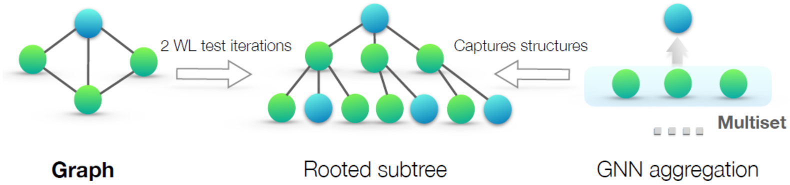
- Representational power of a GNN: when a GNN maps two nodes to the same location (in the embedding space)?

(Overview of) Theoretical Framework



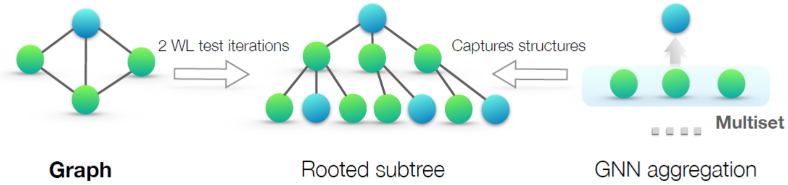
- Representational power of a GNN: when a GNN maps two nodes to the same location (in the embedding space)?
- Maximally powerful GNN: its aggregation scheme must be *injective*!

(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.

(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

1. Introduction
2. Preliminaries
   - GNNs
   - WL test
   - (Overview of) Theoretical Framework
3. Representational capacity of GNNs
4. Graph Isomorphism Network (GIN)
5. Less powerful, but still interesting GNNs
   - 1-Layer Perceptrons
   - GCN
   - GraphSAGE
6. Experiments
7. Summary
8. Future Research
9. Roles and Responsibilities

Representational capacity of GNNs

- Recall that maximally powerful GNN has injective aggregation scheme.

Representational capacity of GNNs

- Recall that maximally powerful GNN has injective aggregation scheme.
- Two (non)isomorphic graphs are mapped to the (different)same representation(s).

Representational capacity of GNNs

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

Representational capacity of GNNs

- Recall that 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.

Representational capacity of GNNs

- Recall that 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"?

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?

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!

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.

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.

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.

Outline

1. Introduction
2. Preliminaries
   - GNNs
   - WL test
   - (Overview of) Theoretical Framework
3. Representational capacity of GNNs
4. Graph Isomorphism Network (GIN)
5. Less powerful, but still interesting GNNs
   - 1-Layer Perceptrons
   - GCN
   - GraphSAGE
6. Experiments
7. Summary
8. Future Research
9. Roles and Responsibilities

Graph Isomorphism Network (GIN)

- Well we've proved (or more accurately, seen) that GNNs under certain conditions is maximally powerful.

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!

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 *i.e.* parametrizing universal multiset functions with neural networks.

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 *i.e.* parametrizing universal multiset functions with neural networks.

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.

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!

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.

Graph Isomorphism Network (GIN)

Lemma 5

Assume $\mathcal{X}$ is countable. There exists a function $f : \mathcal{X} \rightarrow \mathbb{R}^n$ 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.

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. (also, make ϵ a learnable parameter, or a fixed scalar)

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. (also, make ϵ 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.

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.

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.

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.
- To do that, let us use information from all depths/iterations of the model (inspired by Jumping Knowledge Networks (Xu *et al.*, 2018)):

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

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.
- To do that, let us use information from all depths/iterations of the model (inspired by Jumping Knowledge Networks (Xu *et al.*, 2018)):

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

- 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.

Outline

1 Introduction
2 Preliminaries
- GNNs
- WL test
- (Overview of) Theoretical Framework

3 Representational capacity of GNNs
4 Graph Isomorphism Network (GIN)
5 Less powerful, but still interesting GNNs
- 1-Layer Perceptrons
- GCN
- GraphSAGE

6 Experiments
7 Summary
8 Future Research
9 Roles and Responsibilities

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)

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?

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

1. Introduction
2. Preliminaries
   - GNNs
   - WL test
   - (Overview of) Theoretical Framework
3. Representational capacity of GNNs
4. Graph Isomorphism Network (GIN)
5. Less powerful, but still interesting GNNs
   - 1-Layer Perceptrons
   - GCN
   - GraphSAGE
6. Experiments
7. Summary
8. Future Research
9. Roles and Responsibilities

MLP?

- The function f in Lemma 5 helps map distinct multisets to unique embeddings.

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 (Hornik, 1991)

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 (Hornik, 1991)

How powerful is MLP?

Universal Approximation Theorem (Hornik, 1991)

Define

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

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 $\mathcal{N}_k^{(n)}(\psi)$ is dense in $\mathscr{C}(X)$ for all compact subsets X of $\mathbb{R}^k$.*

How powerful is MLP?

Universal Approximation Theorem (Hornik, 1991)

Define

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

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 $\mathcal{N}_k^{(n)}(\psi)$ 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.

How powerful is MLP?

Universal Approximation Theorem (Hornik, 1991)

Define

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

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 $\mathcal{N}_k^{(n)}(\psi)$ 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.

Without MLP?

- Many modern GNNs use a 1-layer perceptron $\sigma \circ W$: a linear mapping followed by a non-linear activation function.

Without MLP?

- Many modern GNNs 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?

Without MLP?

- Many modern GNNs 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, $\sum_{x \in X_1} \mathrm{ReLU}(Wx) = \sum_{x \in X_2} \mathrm{ReLU}(Wx)$

Lemma 7: Contemplation

- 1-layer perceptrons can behave much like linear mappings, so the GNN layers degenerate into simply summing over neighborhood features.

Lemma 7: Contemplation

- 1-layer perceptrons can behave much like linear mappings, so the GNN layers degenerate into simply summing over neighborhood features.
- Proof idea: bias term is lacking in the linear mapping.
- Unlike models using MLPs, 1-layer perceptron (even with the bias term) is *not a universal approximator* of multiset functions.

Lemma 7: Contemplation

- 1-layer perceptrons can behave much like linear mappings, so the GNN layers degenerate into simply summing over neighborhood features.
- Proof idea: bias term is lacking in the linear mapping.
- Unlike models using MLPs, 1-layer perceptron (even with the bias term) is *not a universal approximator* of multiset functions.
- Even if GNNs with 1-layer perceptrons can be injective to some degree, such embeddings may not adequately capture structural similarity. (may be difficult for simple classifiers to fit)

Outline

1. Introduction
2. Preliminaries
   - GNNs
   - WL test
   - (Overview of) Theoretical Framework
3. Representational capacity of GNNs
4. Graph Isomorphism Network (GIN)
5. Less powerful, but still interesting GNNs
   - 1-Layer Perceptrons
   - GCN
   - GraphSAGE
6. Experiments
7. Summary
8. Future Research
9. Roles and Responsibilities

GCN

- As described previously, Graph Convolutional Network (GCN; Kipf & Welling, 2017) 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

- As described previously, Graph Convolutional Network (GCN; Kipf & Welling, 2017) takes the form:

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

- GCN utilizes mean aggregator.

GCN

- As described previously, Graph Convolutional Network (GCN; Kipf & Welling, 2017) takes the form:

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

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

Mean aggregator

- Consider two multisets $X_1 = (S, m)$ and $X_2 = (S, km)$

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

- 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.

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 $k \in \mathbb{N}$.

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

1. Introduction
2. Preliminaries
   - GNNs
   - WL test
   - (Overview of) Theoretical Framework
3. Representational capacity of GNNs
4. Graph Isomorphism Network (GIN)
5. Less powerful, but still interesting GNNs
   - 1-Layer Perceptrons
   - GCN
   - GraphSAGE
6. Experiments
7. Summary
8. Future Research
9. Roles and Responsibilities

GraphSAGE

- As described previously, GraphSAGE (Hamilton *et al.*, 2017) takes the form:

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

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

GraphSAGE

- As described previously, GraphSAGE (Hamilton *et al.*, 2017) takes the form:

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

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

- GraphSAGE utilizes max-pooling aggregator.

GraphSAGE

- As described previously, GraphSAGE (Hamilton *et al.*, 2017) 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?

Max-pooling aggregator

- Unlike previous aggregators, max-pooling can't capture exact structure nor the distribution!

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 $X = (S, m)$

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

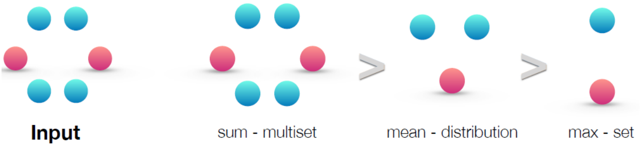
Assume $\mathcal{X}$ is countable. There exists a function $f : \mathcal{X} \to \mathbb{R}^\infty$ so that $h(X) = \max_{x \in X} f(x)$, $h(X_1) = h(X_2)$ *if only if* multisets X_1 and X_2 have the same underlying set.

Summary

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

Summary

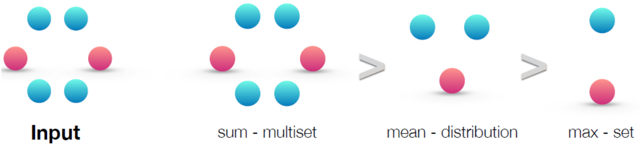
- Let us rank the three aggregators by their representational power:



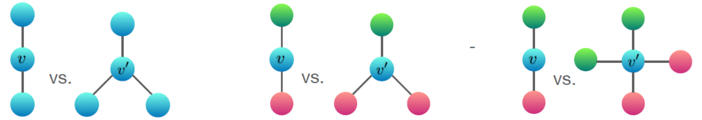
- Sum over multiset aggregator (as in GIN) completely captures the exact structure of graph.

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.

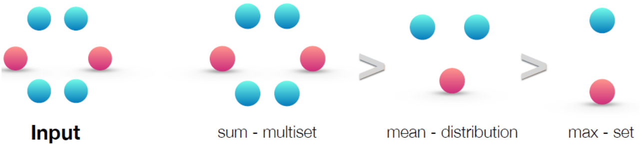


(a) Mean and Max both fail (b) Max fails (c) Mean and Max both fail

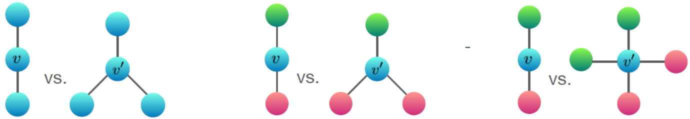
- Mean aggregator (as in GCN) captures the statistical and distributional information of the graph.

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.



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

- 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

1 Introduction
2 Preliminaries
- GNNs
- WL test
- (Overview of) Theoretical Framework

3 Representational capacity of GNNs
4 Graph Isomorphism Network (GIN)
5 Less powerful, but still interesting GNNs
- 1-Layer Perceptrons
- GCN
- GraphSAGE

6 Experiments
7 Summary
8 Future Research
9 Roles and Responsibilities

Experiments

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

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 (Yanardag & Vishwanathan, 2015):
  - 4 bioinformatics datasets (*MUTAG, PTC, NCI1, PROTEINS*)
  - 5 social network datasets (*COLLAB, IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, REDDIT-MULTI5K*)

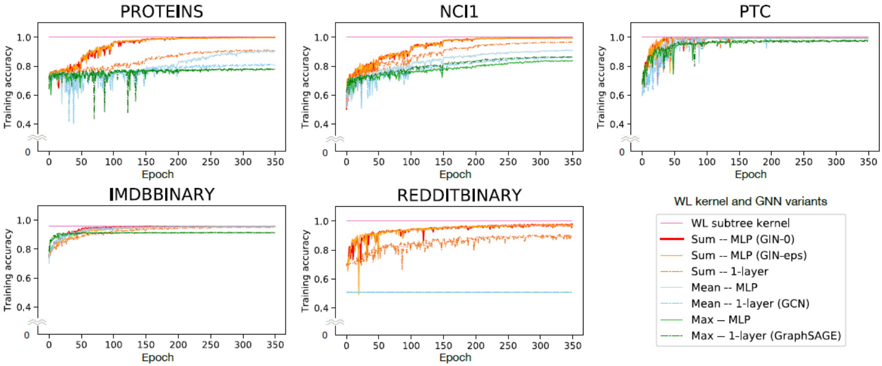
Experiment Design

- 9 graph classification benchmarks were used (Yanardag & Vishwanathan, 2015):
  - 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

Experiment Design

- 9 graph classification benchmarks were used (Yanardag & Vishwanathan, 2015):
  - 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

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	$\mathbf{86.0 \pm 1.8}$ *
	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	$\mathbf{92.6 \pm 4.2}$ *	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**)	$\mathbf{75.1 \pm 5.1}$	$\mathbf{52.3 \pm 2.8}$	$\mathbf{92.4 \pm 2.5}$	$\mathbf{57.5 \pm 1.5}$	$\mathbf{80.2 \pm 1.9}$	$\mathbf{89.4 \pm 5.6}$	$\mathbf{76.2 \pm 2.8}$	$\mathbf{64.6 \pm 7.0}$	$\mathbf{82.7 \pm 1.7}$
	Sum–MLP (**GIN-ϵ**)	$\mathbf{74.3 \pm 5.1}$	$\mathbf{52.1 \pm 3.6}$	$\mathbf{92.2 \pm 2.3}$	$\mathbf{57.0 \pm 1.7}$	$\mathbf{80.1 \pm 1.9}$	$\mathbf{89.0 \pm 6.0}$	$\mathbf{75.9 \pm 3.8}$	63.7 ± 8.2	$\mathbf{82.7 \pm 1.6}$
	Sum–1-Layer	74.1 ± 5.0	$\mathbf{52.2 \pm 2.4}$	90.0 ± 2.7	55.1 ± 1.6	$\mathbf{80.6 \pm 1.9}$	$\mathbf{90.0 \pm 8.8}$	$\mathbf{76.2 \pm 2.6}$	63.1 ± 5.7	82.0 ± 1.5
	Mean–MLP	73.7 ± 3.7	$\mathbf{52.3 \pm 3.1}$	50.0 ± 0.0	20.0 ± 0.0	79.2 ± 2.3	83.5 ± 6.3	75.5 ± 3.4	$\mathbf{66.6 \pm 6.9}$	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

1 Introduction
2 Preliminaries
- GNNs
- WL test
- (Overview of) Theoretical Framework

3 Representational capacity of GNNs
4 Graph Isomorphism Network (GIN)
5 Less powerful, but still interesting GNNs
- 1-Layer Perceptrons
- GCN
- GraphSAGE

6 Experiments
7 Summary
8 Future Research
9 Roles and Responsibilities

Summary

In summary,

Summary

In summary,

- Theoretical foundations for reasoning about the expressive power of GNNS

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)

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*)

Outline

1. Introduction
2. Preliminaries
   - GNNs
   - WL test
   - (Overview of) Theoretical Framework
3. Representational capacity of GNNs
4. Graph Isomorphism Network (GIN)
5. Less powerful, but still interesting GNNs
   - 1-Layer Perceptrons
   - GCN
   - GraphSAGE
6. Experiments
7. Summary
8. **Future Research**
9. Roles and Responsibilities

Future research

- Different aggregators?

Future research

- Different aggregators?
- Go beyond neighborhood aggregation

Future research

- Different aggregators?
- Go beyond neighborhood aggregation
- Understand/improve the generalization properties of GNNs

Future research

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

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

1. Introduction
2. Preliminaries
   - GNNs
   - WL test
   - (Overview of) Theoretical Framework
3. Representational capacity of GNNs
4. Graph Isomorphism Network (GIN)
5. Less powerful, but still interesting GNNs
   - 1-Layer Perceptrons
   - GCN
   - GraphSAGE
6. Experiments
7. Summary
8. Future Research
9. Roles and Responsibilities

Roles and Responsibilities

- Junghyun Lee prepared all the materials.

Roles and Responsibilities

- Junghyun Lee prepared all the materials.
- And as requested, here is a picture of myself:

Thank you for your attention! Any questions?