Xu et al. 2019: How Powerful are Graph Neural Networks?

Minqi Pan

April 20, 2020

How Powerful are Graph Neural Networks?

- ICLR 2019 Oral, Ernest N. Morial Convention Center, New Orleans, May 7th, 2019
- Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka
- MIT, Stanford University

Outline

1. Building Powerful Graph Neural Networks
   - Preliminaries
   - Theoretical Framework: Overview
   - Graph Isomorphism Network (GIN)
   - Graph-level Readout of GIN
2. Less Powerful but Still Interesting GNNs
   - 1-layer Perceptrons are not Sufficient
   - Structures that Confuse Mean and Max-pooling
   - Mean Learns Distributions
   - Max-pooling Learns Sets with Distinct Elements
   - Remarks on Other Aggregators

Outline

1. Building Powerful Graph Neural Networks
   - Preliminaries
   - Theoretical Framework: Overview
   - Graph Isomorphism Network (GIN)
   - Graph-level Readout of GIN
2. Less Powerful but Still Interesting GNNs
   - 1-layer Perceptrons are not Sufficient
   - Structures that Confuse Mean and Max-pooling
   - Mean Learns Distributions
   - Max-pooling Learns Sets with Distinct Elements
   - Remarks on Other Aggregators

Two Tasks

- Given $G = (V, E)$ with X_v for $v \in V$
- Task 1: Node Classification
  - Denote y_v as the label for $v \in V$
  - Learn a representation vector h_v of v such that v's label can be predicted as
$$y_v = f(h_v)$$
- Task 2: Graph Classification
  - Given a set of graphs $\{G_1, \ldots, G_N\} \subset \mathcal{G}$ and their labels $\{y_1, \ldots, y_N\} \subset \mathcal{Y}$
  - Learn a representation vector h_G that helps predict the label of an entire graph:
$$y_G = g(h_G)$$

Graph Neural Networks

- The k-th layer of a GNN is

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

- $h_v^{(k)}$: the feature vector of node v at the k-th iteration/layer
- $h_v^{(0)} = X_v$
- $\mathcal{N}(v)$: a set of nodes adjacent to v
- The choice of $\text{AGGREGATE}^{(k)}(\cdot)$ and $\text{COMBINE}^{(k)}(\cdot)$ in GNNs is crucial

GraphSAGE (Hamilton et al. 2017)

$$a_v^{(k)} = \text{MAX}\left(\left\{\text{ReLU}\left(W \cdot h_u^{(k-1)}\right), \forall u \in \mathcal{N}(v)\right\}\right)$$
$$h_v^{(k)} = \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right)$$

- The COMBINE step could be a concatenation followed by a linear mapping

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

GCN (Kipf & Welling 2017)

$$h_v^{(k)} = \mathsf{ReLU}\Big(W \cdot \mathsf{MEAN}\big\{h_u^{(k-1)}, \forall u \in \mathcal{N}(v) \cup \{v\}\big\}\Big)$$

- the AGGREGATE and COMBINE steps are integrated

Finally

- For node classification
  - the $h_v^{(K)}$ of the final iteration is used for prediction
- For graph classification
  - the READOUT function aggregates node features from the final iteration to obtain the entire graph's representation h_G:

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

  - READOUT can be a simple permutation invariant function such as summation or a more sophisticated graph-level pooling function

Weisfeiler-Lehman Test (1)

- NO polynomial-time algorithm is known for the graph isomorphism problem yet
- Apart from some corner cases, the WL test is an effective and computationally efficient test that DISTINGUISHES a broad class of graphs
- Its 1-dimensional form, "naive vertex refinement", is analogous to neighbor aggregation in GNNs

Weisfeiler-Lehman Test (2)

1. Aggregates the labels of nodes and their neighborhoods
2. Hashes the aggregated labels into unique new labels

Weisfeiler-Lehman Test (3)

- The algorithm decides that two graphs are NON-isomorphic if at some iteration the labels of the nodes between the two graphs differ
- Shervashidze et al 2011 proposed the WL subtree kernel that measures the SIMILARITY between graphs
  - The kernel uses the counts of node labels at different iterations of the WL test as the feature vector of a graph
  - A node's label at the k-th iteration of WL test represents a subtree structure of height k rooted at the node
  - The graph features considered by the WL subtree kernel are essentially counts of different rooted subtrees in the graph

Outline

1. Building Powerful Graph Neural Networks
   - Preliminaries
   - Theoretical Framework: Overview
   - Graph Isomorphism Network (GIN)
   - Graph-level Readout of GIN
2. Less Powerful but Still Interesting GNNs
   - 1-layer Perceptrons are not Sufficient
   - Structures that Confuse Mean and Max-pooling
   - Mean Learns Distributions
   - Max-pooling Learns Sets with Distinct Elements
   - Remarks on Other Aggregators

Definition (1)

A multiset is a generalized concept of a set that allows multiple instances for its elements. More formally, a multiset is a 2-tuple $X = (S, m)$ where S is the underlying set of X that is formed from its distinct elements, and $m : S \to \mathbb{N}_{\geqslant 1}$ gives the multiplicity of the elements.

Outline

1. Building Powerful Graph Neural Networks
   - Preliminaries
   - Theoretical Framework: Overview
   - Graph Isomorphism Network (GIN)
   - Graph-level Readout of GIN
2. Less Powerful but Still Interesting GNNs
   - 1-layer Perceptrons are not Sufficient
   - Structures that Confuse Mean and Max-pooling
   - Mean Learns Distributions
   - Max-pooling Learns Sets with Distinct Elements
   - Remarks on Other Aggregators

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.

Proof.

- Suppose after k iterations, a GNN $\mathcal{A}$ has $\mathcal{A}(G_1) \neq \mathcal{A}(G_2)$ but the WL test cannot decide G_1 and G_2 are NON-isomorphic
- It follows that from iteration 0 to k in the WL test, G_1 and G_2 always have the same collection of node labels
- In particular, because G_1 and G_2 have the same WL node labels for iteration i and $i+1$ for any $i = 0, \ldots, k-1$, G_1 and G_2 have the same collection, i.e. multiset, of WL node labels $\{l_v^{(i)}\}$ as well as the same collection of node neighborhood

$$\left\{\left(l_v^{(i)}, \left\{l_u^{(i)} : u \in \mathcal{N}(v)\right\}\right)\right\}$$

Otherwise, the WL test would have obtained different collections of node labels at iteration $i+1$ for G_1 and G_2 as different multisets get unique new labels

Proof (Cont.)

- The WL test always relabels different multisets of neighboring nodes into different new labels
- We show that on the same graph $G = G_1$ or G_2, if WL node labels $l_v^{(i)} = l_u^{(i)}$, we always have GNN node features $h_v^{(i)} = h_u^{(i)}$ for any iteration i
  - This apparently holds for $i = 0$ because WL and GNN starts with the same node features
  - Suppose this holds for iteration j, if for any $u, v, l_v^{(j+1)} = l_u^{(j+1)}$, then it must be the case that

$$\left(l_v^{(j)}, \left\{l_w^{(j)} : w \in \mathcal{N}(v)\right\}\right) = \left(l_u^{(j)}, \left\{l_w^{(j)} : w \in \mathcal{N}(u)\right\}\right)$$

Proof (Cont.)

- We show that on the same graph $G = G_1$ or G_2, if WL node labels $l_v^{(i)} = l_u^{(i)}$, we always have GNN node features $h_v^{(i)} = h_u^{(i)}$ for any iteration i
  - By our assumption on iteration j, we must have

$$\left(h_v^{(j)}, \left\{h_w^{(j)} : w \in \mathcal{N}(v)\right\}\right) = \left(h_u^{(j)}, \left\{h_w^{(j)} : w \in \mathcal{N}(u)\right\}\right)$$

  - In the aggregation process of the GNN, the same AGGREGATE and COMBINE are applied
  - The same input, i.e. neighborhood features, generates the same output
  - Thus $h_v^{(j+1)} = h_u^{(j+1)}$
  - By induction, if WL node labels $l_v^{(i)} = l_u^{(i)}$, we always have GNN node features $h_v^{(i)} = h_u^{(i)}$ for any iteration i

Proof (Cont.)

- This creates a valid mapping ϕ such that $h_v^{(i)} = \phi(l_v^{(i)})$ for any $v \in G$
- It follows from G_1 and G_2 have the same multiset of WL neighborhood labels that G_1 and G_2 also have the same collection of GNN neighborhood features

$$\left\{\left(h_v^{(i)}, \left\{h_u^{(i)} : u \in \mathcal{N}(v)\right\}\right)\right\} =$$
$$\left\{\left(\phi(l_v^{(i)}), \left\{\phi(l_u^{(i)}) : u \in \mathcal{N}(v)\right\}\right)\right\}$$

- Thus $\left\{h_v^{(i+1)}\right\}$ are the same

Proof (Cont.)

- In particular, we have the same collection of GNN node features

$$h_v^{(k)}$$

for G_1 and G_2

- Because the graph level readout function is permutation invariant with respect to the collection of node features,

$$\mathcal{A}(G_1) = \mathcal{A}(G_2)$$

- Hence we have reached a contradiction



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 graphs G_1 and G_2 that the Weisfeiler-Lehman test of isomorphism decides as non-isomorphic, to different embeddings if the following conditions hold:

1. *$\mathcal{A}$ aggregates and updates node features iteratively with*

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

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

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

Proof.

- Let $\mathcal{A}$ be a GNN where the condition holds
- Let G_1, G_2 be any graphs which the WL test decides as non-isomorphic at iteration K
- Because the graph-level readout function is injective, i.e., it maps distinct multiset of node features into unique embeddings, it suffices to show that $\mathcal{A}$'s neighborhood aggregation process, with sufficient iterations, embeds G_1 and G_2 into different multisets of node features

Proof (Cont.)

- Let us assume $\mathcal{A}$ updates node representations as

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

  with injective functions f and ϕ
- The WL test applies a predetermined injective hash function g to update the WL node labels $l_v^{(k)}$:

$$l_v^{(k)} = g\big(l_v^{(k-1)}, \big\{l_u^{(k-1)} : u \in \mathcal{N}(v)\big\}\big)$$

Proof (Cont.)

- We will show, by induction, that for any iteration k, there always exists an injective function φ such that $h_v^{(k)} = \varphi\big(l_v^{(k)}\big)$
  - This apparently holds for $k = 0$ because the initial node features are the same for WL and GNN $l_v^{(0)} = h_v^{(0)}$ for all $v \in G_1, G_2$; so φ could be the identity function for $k = 0$
  - Suppose this holds for iteration $k - 1$, we show that it also holds for k
  - Substituting $h_v^{(k-1)}$ with $\varphi\big(l_v^{(k-1)}\big)$ gives us

$$h_v^{(k)} = \phi(\varphi\big(l_v^{(k-1)}\big), f\big(\{\varphi\big(l_u^{(k-1)}\big) : u \in \mathcal{N}(v)\}\big)$$

  - Since the composition of injective functions is injective, there exists some injective function ψ so that

$$h_v^{(k)} = \psi\big(l_v^{(k-1)}, \{l_u^{(k-1)} : u \in \mathcal{N}(v)\}\big)$$

Proof (Cont.)

- We will show, by induction, that for any iteration k, there always exists an injective function φ such that $h_v^{(k)} = \varphi\left(l_v^{(k)}\right)$
  - Then we have

$$h_v^{(k)} = \psi \circ g^{-1} g\left(l_v^{(k-1)}, \left\{l_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right) = \psi \circ g^{-1}\left(l_v^{(k)}\right)$$

  - $\varphi = \psi \circ g^{-1}$ is injective because the composition of injective functions is injective
  - Hence for any iteration k, there always exists an injective function φ such that

$$h_v^{(k)} = \varphi\left(l_v^{(k)}\right)$$

Proof (Cont.)

- At the K-th iteration, the WL test decides that G_1 and G_2 are non-isomorphic, that is the multisets $\{l_v^{(K)}\}$ are different for G_1 and G_2
- The GNN $\mathcal{A}$'s node embeddings

$$\{h_v^{(K)}\} = \{\varphi(l_v^{(K)})\}$$

must also be different for G_1 and G_2 because of the injectivity of φ

□

Lemma (4)

Assume the input feature space $\mathcal{X}$ is countable. Let $g^{(k)}$ be the function parameterized by a GNN's k-th layer for $k = 1, \ldots, L$, where $g^{(1)}$ is defined on multisets $X \subset \mathcal{X}$ of bounded size. The range of $g^{(k)}$, i.e., the space of node hidden features $h_v^{(k)}$, is also countable for all $k = 1, \ldots, L$.

Proof.

Sort each multiset X ($|X| < N$) by an enumeration Z of $\mathcal{X}$ and pad it with a dummy element e to an N-tuple over $\mathcal{X} \cup \{e\}$; this map is injective. Finite tuples over a countable set form a countable set (Rudin, *Principles of Mathematical Analysis*, Theorem 2.13), so the multisets, and hence the range of $g^{(1)}$, are countable. Induct on k for $g^{(2)}, \ldots, g^{(L)}$. □

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

Proof.

- We first prove that there exists a mapping f so that $\sum_{x \in X} f(x)$ is unique for each multiset X of bounded size
- Because $\mathcal{X}$ is countable, there exists a mapping $Z : \mathcal{X} \to \mathbb{N}$ from $x \in \mathcal{X}$ to natural numbers
- Because the cardinality of multiset X is bounded, there exists a number $N \in \mathbb{N}$ so that $|X| < N$ for all X
- Then an example of such f is

$$f(x) = N^{-Z(x)}$$

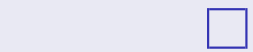
  which can be viewed as a more compressed form of an one-hot vector or N-digit presentation
- Thus $h(X) = \sum_{x \in X} f(x)$ is an injective function of multisets

Proof (Cont.)

- $\phi\big(\sum_{x\in X} f(x)\big)$ is permutation invariant so it is a well-defined multiset function. For any multiset function g, we can construct such ϕ by letting

$$\phi\Big(\sum_{x\in X} f(x)\Big) = g(X)$$

- Note that such ϕ is well-defined because $h(X) = \sum_{x\in X} f(x)$ is injective



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,

$$h(c, X) = (1 + \epsilon) \cdot f(c) + \sum_{x \in X} f(x)$$

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

$$g(c, X) = \varphi\big((1 + \epsilon) \cdot f(c) + \sum_{x \in X} f(x)\big)$$

for some function φ.

Proof.

- We consider

$$f(x) = N^{-Z(x)}$$

where $|X| < N$ for all X and $Z : \mathcal{X} \to \mathbb{N}$ maps from $x \in \mathcal{X}$ to natural numbers

- Let

$$h(c, X) \equiv (1 + \epsilon) \cdot f(c) + \sum_{x \in X} f(x)$$

- Claim: if ϵ is an irrational number, for any $(c', X') \neq (c, X)$ with $c, c' \in \mathcal{X}$ and $X, X' \subset \mathcal{X}$

$$h(c, X) \neq h(c', X')$$

holds

Proof (Cont.)

- We prove by contradiction
- For any (c, X), suppose there exists (c', X') such that $(c', X') \neq (c, X)$ but

$$h(c, X) = h(c', X')$$

  holds
- Let us consider the following two cases
  1. $c' = c$ but $X' \neq X$
  2. $c' \neq c$



Proof (Cont.)

- For the first case: $c' = c$ but $X' \neq X$
  - $h(c, X) = h(c, X')$ implies

$$\sum_{x \in X} f(x) = \sum_{x \in X'} f(x)$$

  - It follows from Lemma 5 that the equality will not hold, because with $f(x) = N^{-Z(x)}$,

$$X' \neq X \implies \sum_{x \in X} f(x) \neq \sum_{x \in X'} f(x)$$

  - Thus, we reach a contradiction



Proof (Cont.)

- For the second case: $c' \neq c$
  - We can similarly rewrite $h(c, X) = h(c', X')$ as

$$\epsilon \cdot (f(c) - f(c')) = \Big(f(c') + \sum_{x \in X'} f(x)\Big) - \Big(f(c) + \sum_{x \in X} f(x)\Big)$$

  - Because ϵ is an irrational number and $f(c) - f(c')$ is a non-zero rational number, L.H.S. is irrational
  - R.H.S. is rational because the sum of a finite number of rational numbers
  - Thus, we reach a contradiction



Proof (Cont.)

- For any function g over the pairs (c, X), we can construct such φ for the desired decomposition by letting

$$\varphi\big((1+\epsilon)\cdot f(c) + \sum_{x\in X} f(x)\big) = g(c, X)$$

- Note that such φ is well-defined because

$$h(c, X) = (1+\epsilon)\cdot f(c) + \sum_{x\in X} f(x)$$

  is injective



- GIN uses MLP to learn f and φ
- In practice, we model

$$f^{(k+1)} \circ \varphi^{(k)}$$

with a single MLP:

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

Outline

1. Building Powerful Graph Neural Networks
   - Preliminaries
   - Theoretical Framework: Overview
   - Graph Isomorphism Network (GIN)
   - Graph-level Readout of GIN
2. Less Powerful but Still Interesting GNNs
   - 1-layer Perceptrons are not Sufficient
   - Structures that Confuse Mean and Max-pooling
   - Mean Learns Distributions
   - Max-pooling Learns Sets with Distinct Elements
   - Remarks on Other Aggregators

- An important aspect of the graph-level readout is that node representations, corresponding to subtree structures, get more refined and global as the number of iterations increases
- A sufficient number of iterations is key to achieving good discriminative power
- Yet, features from earlier iterations may sometimes generalize better
- To consider all structural information, we use information from all depths/iterations of the model, concatenating graph representations across all iterations/layers of GIN:

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

Outline

1. Building Powerful Graph Neural Networks
   - Preliminaries
   - Theoretical Framework: Overview
   - Graph Isomorphism Network (GIN)
   - Graph-level Readout of GIN
2. Less Powerful but Still Interesting GNNs
   - 1-layer Perceptrons are not Sufficient
   - Structures that Confuse Mean and Max-pooling
   - Mean Learns Distributions
   - Max-pooling Learns Sets with Distinct Elements
   - Remarks on Other Aggregators

- The function f in Lemma 5 helps map distinct multisets to unique embeddings
- f can be parameterized by an MLP by the universal approximation theorem
- Nonetheless, many existing GNNs instead use a 1-layer perceptron

  $$\sigma \circ W,$$

  a linear mapping followed by a non-linear activation function such as a ReLU
- Such 1-layer mappings are examples of Generalized Linear Models
- Therefore, we are interested in understanding whether 1-layer perceptrons are enough for graph learning

Lemma (7)

There exists finite multisets $X_1 \neq X_2$ so that for any linear mappings W,

$$\sum_{x \in X_1} ReLU(Wx) = \sum_{x \in X_2} ReLU(Wx).$$

Proof.

- Let us consider the example

$$X_1 = \{1, 1, 1, 1, 1\},$$
$$X_2 = \{2, 3\},$$

  i.e. two different multisets of positive numbers that sum up to the same value
- We will be using the homogeneity of ReLU

Proof (Cont.)

- Let W be an arbitrary linear transform that maps $x \in X_1, X_2$ into $\mathbb{R}^n$
- It is clear that, at the same coordinates, Wx are either positive or negative for all x because all x in X_1 and X_2 are positive
- It follows that ReLU(Wx) are either positive or 0 at the same coordinate for all x in X_1, X_2
- For the coordinates where ReLU(Wx) are 0, we have

$$\sum_{x \in X_1} \text{ReLU}(Wx) = \sum_{x \in X_2} \text{ReLU}(Wx)$$

Proof (Cont.)

- For the coordinates where Wx are positive, linearity still holds. It follows from linearity that

$$\sum_{x \in X} \text{ReLU}(Wx) = \text{ReLU}\big(W \sum_{x \in X} x\big)$$

  where X could be X_2 or X_1
- Because $\sum_{x \in X_1} x = \sum_{x \in X_2} x$, we have the following as desired

$$\sum_{x \in X_1} \text{ReLU}(Wx) = \sum_{x \in X_2} \text{ReLU}(Wx)$$



Outline

1. Building Powerful Graph Neural Networks
   - Preliminaries
   - Theoretical Framework: Overview
   - Graph Isomorphism Network (GIN)
   - Graph-level Readout of GIN
2. Less Powerful but Still Interesting GNNs
   - 1-layer Perceptrons are not Sufficient
   - Structures that Confuse Mean and Max-pooling
   - Mean Learns Distributions
   - Max-pooling Learns Sets with Distinct Elements
   - Remarks on Other Aggregators

- What happens if we replace the sum in

$$h(X) = \sum_{x \in X} f(x)$$

  with mean or max-pooling as in GCN and GraphSAGE?
- Mean and max-pooling aggregators are still well-defined multiset functions because they are permutation invariant
- But they are NOT injective
- Sum captures the full multiset, mean only its proportions (distribution), max only its underlying set
- Neighbor multisets they confuse:
  - $\{a, a\}$ vs. $\{a, a, a\}$: mean and max fail; sum works
  - $\{a, b\}$ vs. $\{a, b, b\}$: max fails
  - $\{a, b\}$ vs. $\{a, a, b, b\}$: mean and max fail

Outline

1. Building Powerful Graph Neural Networks
   - Preliminaries
   - Theoretical Framework: Overview
   - Graph Isomorphism Network (GIN)
   - Graph-level Readout of GIN
2. Less Powerful but Still Interesting GNNs
   - 1-layer Perceptrons are not Sufficient
   - Structures that Confuse Mean and Max-pooling
   - Mean Learns Distributions
   - Max-pooling Learns Sets with Distinct Elements
   - Remarks on Other Aggregators

- To characterize the class of multisets that the mean aggregator can distinguish, consider the example

$$X_1 = (S, m)$$
$$X_2 = (S, k \cdot m)$$

  where X_1 and X_2 have the same set of distinct elements, but X_2 contains k copies of each element of X_1
- Any mean aggregator maps X_1 and X_2 to the same embedding, because it simply takes averages over individual element features
- Thus the mean captures the distribution (proportions) of elements in a multiset, but not the exact multisets

Corollary (8)

Assume $\mathcal{X}$ is countable. There exists a function $f : \mathcal{X} \to \mathbb{R}^n$ so that for $h(X) = \frac{1}{|X|} \sum_{x \in X} f(x)$,

$$h(X_1) = h(X_2) \Leftrightarrow X_1, X_2 \text{ have the same distribution.}$$

That is, assuming $|X_2| \geqslant |X_1|$, we have $X_1 = (S, m)$ and $X_2 = (S, k \cdot m)$ for some $k \in \mathbb{N}_{\geqslant 1}$.

Proof.

- Suppose multisets X_1 and X_2 have the same distribution, without loss of generality, let us assume $X_1 = (S, m)$ and $X_2 = (S, k \cdot m)$ for some $k \in \mathbb{N}_{\geqslant 1}$, i.e. X_1 and X_2 have the same underlying set and the multiplicity of each element in X_2 is k times of that in X_1
- Then we have

$$|X_2| = k|X_1|$$

$$\sum_{x \in X_2} f(x) = k \cdot \sum_{x \in X_1} f(x)$$

$$\frac{1}{|X_2|} \sum_{x \in X_2} f(x) = \frac{1}{k \cdot |X_1|} \cdot k \cdot \sum_{x \in X_1} f(x)$$

Proof (Cont.)

- Now we show that there exists a function f so that $\frac{1}{|X|}\sum_{x\in X} f(x)$ is unique for distributionally equivalent X
- Because $\mathcal{X}$ is countable, there exists a mapping $Z : \mathcal{X} \to \mathbb{N}$ from $x \in \mathcal{X}$ to natural numbers
- Because the cardinality of multisets X is bounded, there exists a number $N \in \mathbb{N}$ such that $|X| < N$ for all X
- Then an example of such f is

$$f(x) = N^{-2Z(x)}$$

□

- The mean aggregator may perform well if, for the task, the statistical and distributional information in the graph is more important than the exact structure
- Moreover, when node features are diverse and rarely repeat, the mean aggregator is as powerful as the sum aggregator
- This may explain why GNNs with mean aggregators are effective for node classification tasks, such as classifying article subjects and community detection, where node features are rich and the distribution of the neighborhood features provides a strong signal for the task

Outline

1. Building Powerful Graph Neural Networks
   - Preliminaries
   - Theoretical Framework: Overview
   - Graph Isomorphism Network (GIN)
   - Graph-level Readout of GIN
2. Less Powerful but Still Interesting GNNs
   - 1-layer Perceptrons are not Sufficient
   - Structures that Confuse Mean and Max-pooling
   - Mean Learns Distributions
   - Max-pooling Learns Sets with Distinct Elements
   - Remarks on Other Aggregators

- Max-pooling considers multiple nodes with the same feature as only one node (i.e., treats a multiset as a set)
- Max-pooling captures neither the exact structure nor the distribution
- However, it may be suitable for tasks where it is important to identify representative elements or the "skeleton", rather than to distinguish the exact structure or distribution

Corollary (9)

Assume $\mathcal{X}$ is countable. Then there exists a function $f : \mathcal{X} \to \mathbb{R}^\infty$ so that for $h(X) = \max_{x \in X} f(x)$,

$$h(X_1) = h(X_2) \Leftrightarrow X_1, X_2 \text{ have the same underlying set}$$

Proof.

- Suppose multisets X_1 and X_2 have the same underlying set S, then we have

$$\max_{x \in X_1} f(x) = \max_{x \in S} f(x) = \max_{x \in X_2} f(x)$$

- Now we show that there exists a mapping f so that $\max_{x \in X} f(x)$ is unique for X's with the same underlying set
- Because $\mathcal{X}$ is countable, there exists a mapping $Z : \mathcal{X} \to \mathbb{N}$
- Then an example of such $f : \mathcal{X} \to \mathbb{R}^\infty$ is defined as $f_i(x) = 1$ for $i = Z(x)$ and $f_i(x) = 0$ otherwise, where $f_i(x)$ is the i-th coordinate of $f(x)$; such an f essentially maps a multiset to its one-hot embedding



Outline

1. Building Powerful Graph Neural Networks
   - Preliminaries
   - Theoretical Framework: Overview
   - Graph Isomorphism Network (GIN)
   - Graph-level Readout of GIN
2. Less Powerful but Still Interesting GNNs
   - 1-layer Perceptrons are not Sufficient
   - Structures that Confuse Mean and Max-pooling
   - Mean Learns Distributions
   - Max-pooling Learns Sets with Distinct Elements
   - Remarks on Other Aggregators

- There are other non-standard neighbor aggregation schemes that we do not cover
- E.g. weighted average via attention
- E.g. LSTM pooling
- We emphasize that our theoretical framework is general enough to characterize the representational power of any aggregation-based GNNs
- In the future, it would be interesting to apply our framework to analyze and understand other aggregation schemes