Two papers on GNN theory: How Power are GNN and Deeper Insight of GCN Semisupervised

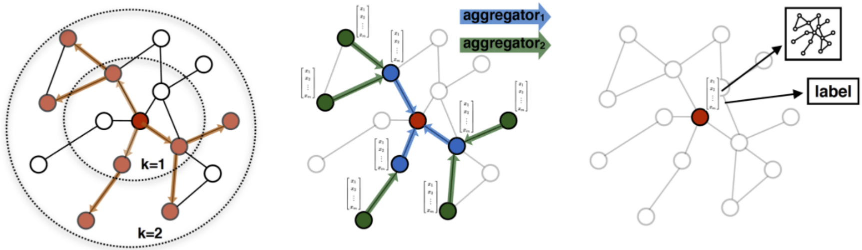
Presenter: Ji Gao
https://qdata.github.io/deep2Read

June 3, 2019

Outline

1. Review: Graph Neural Networks
2. How powerful are graph neural networks?
   - Graph Neural Network
   - Graph Isomorphism and Weisfeiler-Lehman Test
   - Graph Isomorphic Network(GIN)
   - Experiment result
3. Deeper Insights into Graph Convolutional Networks for semi-supervised learning
   - Issues of GCN
   - Algorithm
   - Experiment result
4. Reference

Graph Neural Network



1. Sample neighborhood

2. Aggregate feature information from neighbors

3. Predict graph context and label using aggregated information

How powerful are graph neural networks?

How powerful are graph neural networks? *Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka*, ICLR 2019 [XHLJ18]

- GNN is weaker than Weisfeiler-Lehman graph isomorphism test
- GNN cannot learn to distinguish certain types
- Propose an architecture that is provably as powerful as WL test.

GNN definition

GNN layer

The k-th layer of a GNN is:

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

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

GraphSage - Maxpooling variant[HYL17]

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

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

GCN[KW16]

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

Graph classification via GNN

- For graph classification, an additional step for GNN is to combine the embedding of all the nodes together:

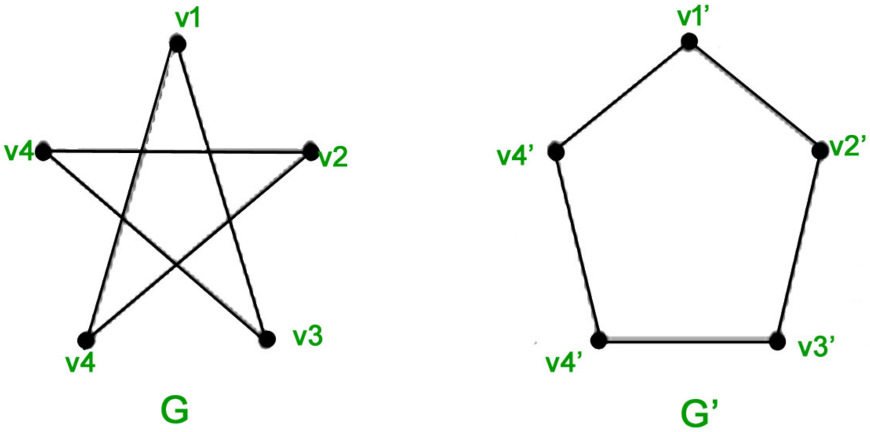
$$h_G = \mathrm{READOUT}\big(\{h_v^{(K)} \mid v \in G\}\big).$$

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

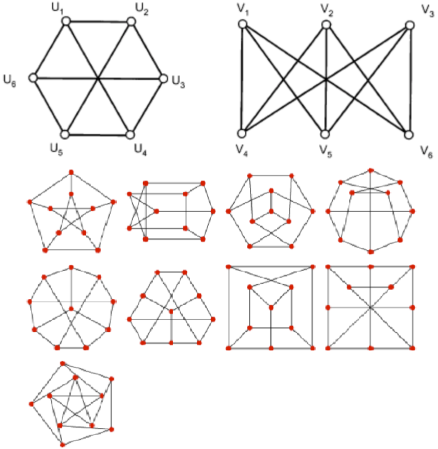
Graph Isomorphism

- **Graph Isomorphism:** Graph $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic graphs if a permutation p of V_1 makes G_1 equals G_2: Any $(u, v) \in E_1$ has $(p(u), p(v)) \in E_2$
- Verify graph isomorphism is NP, somewhere between P and NP-Complete.

Graph Isomorphism Example



Graph Isomorphism Example



Weisfeiler-Lehman Test

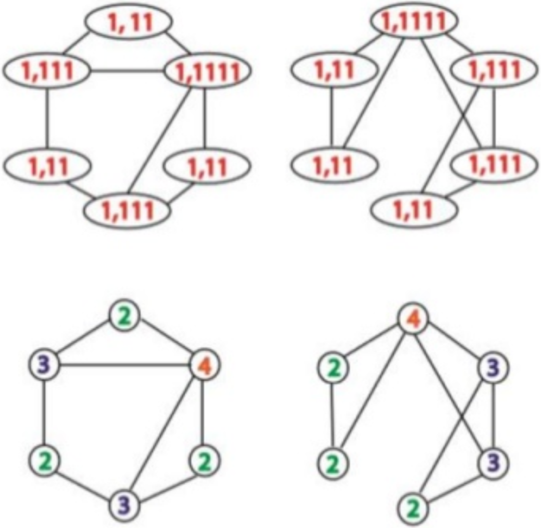
- Weisfeiler-Lehman Test is a subtree-based method to solve graph-isomorphic problem.

Weisfeiler-Lehman Test

- Initialize all nodes with some node features, or with same label 1.
- Iteratively:
  1. Aggregates the labels of nodes and their neighborhoods.
  2. Hashes the aggregated labels into unique new labels.

Weisfeiler-Lehman Test

From https://www.slideshare.net/pratikshukla11/graph-kernelpdf:



WL Test and GNN

- In the GraphSage paper, the authors claim that WL Test is one special case of GraphSage, only the hashing algorithm is replaced with trainable aggregation function.
- However, as shown in this paper, common aggregators such as mean-pooling/max-pooling is weaker than the hash function.

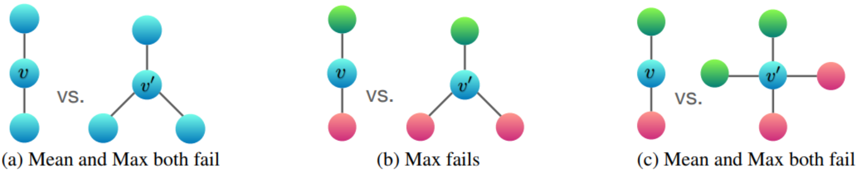


Figure 3: **Examples of graph structures that mean and max aggregators fail to distinguish.** Between the two graphs, nodes v and v' get the same embedding even though their corresponding graph structures differ. Figure 2 gives reasoning about how different aggregators "compress" different multisets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

WL Test is stronger than GNN

- Lemma: Let G_1 and G_2 be any two non-isomorphic graphs. If a graph neural network $A : G \to \mathcal{R}^D$ 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.
- *(Assume the hashing function is extremely powerful.)*

How to make GNN powerful?

Theorem

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:

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

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

where the functions f, which operates on multisets, and ϕ are injective (*1-1 function*).

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

- Conclusion: The problem is in the aggregation function (and the readout function).

How to make GNN powerful?

- Idea is simple and straightforward, however, sum/max-pooling/mean-pooling are not injective on multiset.
- The authors model the aggregator with a neural network:

Graph Isomorphism Network(GIN)

A GIN layer is defined as:

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

GIN Guarantees

Theorem

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.

Lemma

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 $g(c, X) = \varphi\left((1+\epsilon) \cdot f(c) + \sum_{x \in X} f(x)\right)$ for some function φ.

By these two conditions, GIN is as generalize as WL test with a proper MLP ϕ learned and a proper ϵ choose.

Graph classification Readout of GIN

- GIN use a readout that loads the input on every layers:

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

Study on the variants

- 1-layer perceptions are not sufficient.
- Mean and Max-pooling are not good enough (With counter-examples above).
- Mean pooling is better for tasks with diverse and rarely repeat features, and max pooling is better for catching a general relationship.

Experiment

- Use 9 datasets
- **Methods in comparison**:
  - GIN-0: ϵ is always 0
  - GIN-ϵ: ϵ is also updated by gradient in the training
  - SVM
  - DCNN
  - DGCNN
  - AWL
  - GraphSage-Mean
  - GCN

Experiment result

	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

Table 1: **Test set classification accuracies (%).** The best-performing GNNs are highlighted with boldface. On datasets where GINs' accuracy is not strictly the highest among GNN variants, we see that GINs are still comparable to the best GNN because a paired t-test at significance level 10% does not distinguish GINs from the best; thus, GINs are also highlighted with boldface. If a baseline performs significantly better than all GNNs, we highlight it with boldface and asterisk.

Deeper Insights into Graph Convolutional Networks for semi-supervised learning

Deeper Insights into Graph Convolutional Networks forsemi-supervised learning *Qimai Li, Zhichao Han, Xiaoming Wu, AAAI 2018* [LHW18]

- GCN performance doesn't increase with number of layers, which contradicts its design logic
- To overcome this problem(Stay with shallow architecture), use co-training and self-training to improve its performance.

GCN formula

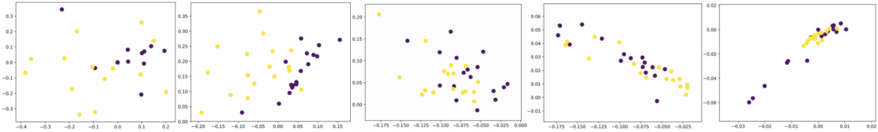
GCN

A GCN layer is defined as:

$$H^{(k+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(k)}W^{(k)})$$

- GCN adds a self-loop to every nodes in the graph.
- The hidden vector of a node is updated by a weighted average of itself and its neighbors.
- Laplacian smoothing

When GCN fails



(a) 1-layer (b) 2-layer (c) 3-layer (d) 4-layer (e) 5-layer

Figure 2: Vertex embeddings of Zachary's karate club network with GCNs with 1,2,3,4,5 layers.

- Generate a 2 dimension vector on every vertex on a small dataset
- Repeatedly apply Laplacian smoothing to the graph will cause it mix together again
- Theorem: The result of Laplacian smoothing will converge to $D^{-\frac{1}{2}}\mathbf{1}\theta$ if the graph has no bipartite components.

Issues of GCN

- GCN used in [KW16] has only 2 layers. However, GCN suppose to be a localize filter, which means it doesn't propagate the information to the whole graph.
- Example: If you don't have a near neighbor with label, GCN won't work.
- Authors also claim that "GCN relies on additional large validation set to select the model, as hyperparameter is crucial."

Co-train a GCN with Random Walk Model

- Idea: Random Walk can explore global graph structure, which is complementary to the GCN result.
- In particular, use random walk model to provide extra labels for GCN to make prediction.
- Calculate the random walk absorbing probabilities matrix
- Algorithm:

Algorithm 1 Expand the Label Set via ParWalks

$P := (L + \alpha\Lambda)^{-1}$
for each class k **do**
  $\boldsymbol{p} := \sum_{j \in \mathcal{S}_k} P_{:,j}$
  Find the top t vertices in $\boldsymbol{p}$
  Add them to the training set with label k
end for

GCN self-training

- Train it again

Algorithm 2 Expand the Label Set via Self-Training

1: $\boldsymbol{Z} := GCN(X) \in \mathbb{R}^{n \times F}$, the output of GCN
2: **for** each class k **do**
3: Find the top t vertices in $Z_{i,k}$
4: Add them to the training set with label k
5: **end for**

Experiment

- **Data:** Cora, Citeseer and Pubmed
- **Methods in comparison:**
  - Label propagation via random walk(LP)
  - ChebyNet
  - GCN: With or without validation set.
  - Co-training
  - Self-training
  - Co-training set union/intersect Self-training set

Experiment Result

Table 3: Classification Accuracy On Cora

	Cora					
Label Rate	0.5%	1%	2%	3%	4%	5%
LP	56.4	62.3	65.4	67.5	69.0	70.2
Cheby	38.0	52.0	62.4	70.8	74.1	77.6
GCN-V	42.6	56.9	67.8	74.9	77.6	79.3
GCN+V	50.9	62.3	72.2	76.5	78.4	79.7
Co-training	56.6	66.4	73.5	75.9	78.9	80.8
Self-training	53.7	66.1	73.8	77.2	79.4	80.0
Union	**58.5**	**69.9**	**75.9**	**78.5**	**80.4**	**81.7**
Intersection	49.7	65.0	72.9	77.1	79.4	80.2

Table 4: Classification Accuracy on CiteSeer

	CiteSeer					
Label Rate	0.5%	1%	2%	3%	4%	5%
LP	34.8	40.2	43.6	45.3	46.4	47.3
Cheby	31.7	42.8	59.9	66.2	68.3	69.3
GCN-V	33.4	46.5	62.6	66.9	68.4	69.5
GCN+V	43.6	55.3	64.9	67.5	68.7	69.6
Co-training	**47.3**	55.7	62.1	62.5	64.5	65.5
Self-training	43.3	58.1	68.2	69.8	70.4	71.0
Union	46.3	**59.1**	66.7	66.7	67.6	68.2
Intersection	42.9	**59.1**	**68.6**	**70.1**	**70.8**	**71.2**

Table 5: Classification Accuracy On PubMed

	PubMed			
Label Rate	0.03%	0.05%	0.1%	0.3%
LP	61.4	66.4	65.4	66.8
Cheby	40.4	47.3	51.2	72.8
GCN-V	46.4	49.7	56.3	76.6
GCN+V	60.5	57.5	65.9	77.8
Co-training	**62.2**	**68.3**	**72.7**	78.2
Self-training	51.9	58.7	66.8	77.0
Union	58.4	64.0	70.7	**79.2**
Intersection	52.0	59.3	69.4	77.6

Experiment result II

Table 6: Accuracy under 20 Labels per Class

Method	CiteSeer	Cora	Pubmed
ManiReg	60.1	59.5	70.7
SemiEmb	59.6	59.0	71.7
LP	45.3	68.0	63.0
DeepWalk	43.2	67.2	65.3
ICA	69.1	75.1	73.9
Planetoid	64.7	75.7	77.2
GCN-V	68.1	80.0	78.2
GCN+V	68.9	80.3	**79.1**
Co-training	64.0	79.6	77.1
Self-training	67.8	80.2	76.9
Union	65.7	**80.5**	78.3
Intersection	**69.9**	79.8	77.0

Reference I

- Will Hamilton, Zhitao Ying, and Jure Leskovec, *Inductive representation learning on large graphs*, Advances in Neural Information Processing Systems, 2017, pp. 1024–1034.
- Thomas N Kipf and Max Welling, *Semi-supervised classification with graph convolutional networks*, arXiv preprint arXiv:1609.02907 (2016).
- Qimai Li, Zhichao Han, and Xiao-Ming Wu, *Deeper insights into graph convolutional networks for semi-supervised learning*, Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
- Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka, *How powerful are graph neural networks?*, arXiv preprint arXiv:1810.00826 (2018).