Introduction to Graph Neural Networks and Their Expressive Power

Tommy He

Abstract

In this report, we will introduce invariant and equivariant graph neural networks (GNN), its general variants and uses, as well as how it works and representations used for them. Then, we will discuss 3 popular specific archetectures, namely linear GNNs, message passing GNNs, and folklore GNNs, with their constructions. In the end, we will include current theoretical results about how well GNNs can classify graphs and approximate arbitrary functions.

1 Introduction to Graph Neural Networks

Why do we use GNNs? They allow us to encode more information and relationship between data and naturally appear in a wide variety of problems such as molecules in chemistry [3] or social networks. Graphs can in fact be viewed as generalizations of datasets by encapsulating each i.i.d. sample as a vertex with no edges. In this sense, GNNs are generalizations of MLPs/CNNs but allow for relationships between samples as well [5]. Generally, our task can be categorizes into 3 catagories: node-level, edge-level, and graph-level.

- *Node-level* tasks deal with classification or predictions with individual nodes. This might include a social network where we attempt to classify which individuals, which are modelled as the nodes, form a friend group or predicting properties of an atom in a molecule.
- *Edge-level* tasks are related to edge classification or predictionn. This might include determining relationships between individuals or objects in an image.
- *Graph-level* tasks are ones related to properties of the graph as a whole. For instance, predicting how a molecule smells or the dynamics of a social group.

A GNN is an optimizable transformation on the attributes of a graph, such as the nodes, edges, and global information that is invariant, if it should be independent on node ordering, or equivariant, if the output should be permuted similarly to the input, under permutations [7]. The reason for this is that oftentimes we want to learn a representation that is either independent of the nodes or preserved a permutation of them. GNNs take in a graph with information loaded

in its structure as features and progressively transforms these embeddings without changing the connectivity. Note that simply using the adjacency matrix directly as input would not work since it depends on the arbitrary ordering of the nodes. Concretely for a matrix representation A of a graph, we want either

$$f(\sigma A) = f(A) \text{ invariant}$$
$$f(\sigma A) = Pf(A) \text{ equivariant}$$

for any function f, such as the GNN, where σ is a permutation action on the vertices. Note that composing equivariant function with an invariant function gives an invariant function.

2 Representations of Graphs

Following the notation of [1], let $G = (V, E)$ be a graph with $|V| = n$ and $[n] := \{1, \cdots, n\}$. Let $\mathbb{F}, \mathbb{F}_0\mathbb{F}_{1/2}, \mathbb{F}_1, \mathbb{F}_{1+1/2}$ be arbitrary finite dimensional spaces $\mathbb{R}^p$ used to denote the space of features where $\mathbb{F}_l$ corresponds to the l-th layer. Graphs with embedded features are typically represented in 2 ways. First way, we can view graphs as tensors of order k as $G \in \mathbb{F}^{n^k}$ where the graph may be embedded in an adjacency matrix of dimension n^k with elements as features in $\mathbb{F}$. As a tensor, let G_i for $i \in [n]^k$ represent the feature corresponding to that coordinate. Second way, we can view graphs as their discrete strucuture $G = (V, E)$ along with a feature vector encoding all the features. Let $\mathcal{G}_n$ be the set of unweighted graphs $G = (V, E)$ where $V = [n], E \subseteq V^2$. Then a $G \in \mathcal{G}_n$ along with $h \in \mathbb{F}^n$ is a graph with features on the n vertices.

3 GNN Architectures

We will define the main architectures studied here including message passing GNNs, linear GNNs, and Folklore GNNs. In all of these, there will be a main building block given by a map

$$\mathbb{F}_t^{n^k} \to \mathbb{F}_{t+1}^{n^k}$$

where $\mathbb{F}_t^{n^k}$ is the space of graph representations at layer t, of which there will be T in total. The beginning part of the GNN is built from a composition of these layer maps. Each type of GNN also have an invariant and equivariant version, which is decided by one layer after the initial T building blocks. Finally, all invariant and equivariant versions share a final continuous map

$$m_I : \mathbb{F}_{T+1} \to \mathbb{F} \text{ invariant}$$
$$m_E : \mathbb{T}_{T+1}^n \to \mathbb{F}^n \text{ equivariant.}$$

Essentially, the beginning part of the GNN will take in $G \in \mathbb{F}_h^{n^2}$ and produce a graph embedding in $\mathbb{F}_{T+1}$ if it invariant else an embedding in $\mathbb{F}_{T+1}^n$ if it is equivariant. Then, by passing that result

through m_I or m_E, it will give a feature in $\mathbb{F}$ or $\mathbb{F}^n$.

Linear Graph Neural Networks

Linear Graphs Neural Networks (LGNN) due to [6] is perhaps the most straightforward GNN as it consists of applying transformations directly onto the features after lifting the order of the graph tensor. Take graph $G \in \mathbb{F}_0^{n^k}$. Each layer is given by the map

$$F : \mathbb{F}_l^{n^k} \to \mathbb{F}_{l+1}^{n^k}, G \mapsto f(L(G))$$

where $L : \mathbb{F}_l^{n^k} \to \mathbb{F}_l^{n^k}$ is a linear equivariant function, and $f : \mathbb{F}_l \to \mathbb{F}_{l+1}$ is lernable. For invariance and equivariance, define S^k to be the sum of the features of the kth layer given by

$$S^k : \mathbb{F}^{n^k} \to \mathbb{F}, G \mapsto \sum_{i \in [n]^k} G_i,$$

which is of course invariant since the sum is independent of the ordering. Let us define S_1^k to be a sum of features of the kth layer dependent on the ordering given by

$$S_1^k : \mathbb{F}^{n^k} \to \mathbb{F}^n, G \mapsto \left\{ \sum_{1 \le i_2 \le \cdots \le i_k \le n} G_{i,i_2,\cdots,i_k} \mid i \in [n] \right\}.$$

We also need a map $I^k : \mathbb{F}_0^{n^2} \to \mathbb{F}_1^{n^k}$ to lift the input graph of order 2 to a order k tensor, which can be created as in [6]. With these, we have the 2 types of LGNNs as

$$k\text{-LGNN}_I = \{m_I \circ S^k \circ F_T \circ \cdots \circ F_2 \circ F_1 \circ I^k \mid T \in \mathbb{N}\}$$
$$k\text{-LGNN}_E = \{m_E \circ S_1^k \circ F_T \circ \cdots \circ F_2 \circ F_1 \circ I^k \mid T \in \mathbb{N}\}.$$

where each F_t is defined the same as F above.

Message Passing Graph Neural Networks

Message passing graph neural networks (MPNN) due to [3] works by aggregating information from neighboring nodes to each node to use the graph's connectivity. Normally, they are simply aggregated via a summation and then passed through an update function that is learnable. Suppose we are working with graph $G = (V, E)$ along with features $h^0 \in \mathbb{F}^n$ where $h_i^l \in \mathbb{F}_l$ denote the features of node i in layer l. We update the features every layer as

$$h_i^{l+1} = f_0\left(h_i^l, \sum_{j \sim i} f_1(h_i^l, h_j^l)\right)$$

where $j \sim i$ denotes the nodes j that are neighbors of i in G, and $f_0 : \mathbb{F}_l \times \mathbb{F}_{l+1/2} \to \mathbb{F}_{l+1}$, $f_1 : \mathbb{F}_l \times \mathbb{F}_l \to \mathbb{F}_{l+1/2}$ are learnable. Essentially the features of the next node are updated using the previous features of the same node as well as the sum of its neighbors.

Then, S^1 is invariant same as the LGNN case, and $\mathrm{Id} + \lambda S^1$ is equivariant, which we can see by

$$\begin{aligned}(\mathrm{Id} + \lambda S^1)(\sigma G) &= \sigma G + \lambda S^1(G) \\ &= \sigma(G + \lambda S^1(G)) \\ &= \sigma(\mathrm{Id} + \lambda S^1)(G).\end{aligned}$$

With these, we can define the 2 types of MPGNNs as

$$\begin{aligned}\mathrm{MGNN}_I &= \{m_I \circ S^1 \circ F_T \circ \cdots \circ F_2 \circ F_1 \mid T \in \mathbb{N}\} \\ \mathrm{MGNN}_E &= \{m_E \circ (\mathrm{Id} + \lambda S^1) \circ F_T \circ \cdots \circ F_2 \circ F_1 \mid T \in \mathbb{N}\}.\end{aligned}$$

where each F_t is defined as the message passing layers above $h^l \mapsto h^{l+1}$.

Folklore Graph Neural Networks

Folkore Graph Neural Networks (FGNN) work by lifting the graph tensor into order k similar to LGNN bu then applies a folklore graph layer (FGL), which proves to be very expressive. an FGL layer is defined by

$$F : \mathbb{F}_l^{n^k} \to \mathbb{F}_{l+1}^{n^k}, G \mapsto \left\{ f_0\left(G_i, \sum_{j=1}^{b} \prod_{w=1}^{k} f_w(G_{i_1,\cdots,i_{w-1},j,i_{w+1},\cdots,i_k}) \mid i \in [n]^k \right) \right\}$$

for learnable functions $f_0 : \mathbb{F}_l \times \mathbb{F}_{l1/2} \to \mathbb{F}_{l+1}$ and $f_w : \mathbb{F}_l \to \mathbb{F}_{l+1/2}$ for $w \in [k]$. As shown in [1], FGL is indeed equivariant and indeed very expressive. Similar to in LGNNs, we get the FGNNs to be

$$\begin{aligned}k\text{-FGNN}_I &= \{m_I \circ S^k \circ F_T \circ \cdots \circ F_2 \circ F_1 \circ I^k \mid T \in \mathbb{N}\} \\ k\text{-FGNN}_E &= \{m_E \circ S_1^k \circ F_T \circ \cdots \circ F_2 \circ F_1 \circ I^k \mid T \in \mathbb{N}\}.\end{aligned}$$

after lifting to k order tensors and using S^k, S_1^k for invariance and equivariance where each F_t is defined as F above.

4 Theoretical Results

Graph isomorphism is the problem of finding whether two graphs are isomorphic i.e. there is a bijection of the vertices that preserve the edges. It remains one of the most important

problems to find an efficient solution to. The Weisfeiler-Lehman algorithm is a classical algorithm that can classify a large set of graphs up to isomorphism based on color refinement [4]. It assigns a color per each node and continues to refine the coloring by aggregating the colors of neighboring vertices and stops once it reaches a stable coloring. If two graphs are distinct, then they will have different colorings of course. k-WL is a generalization of WL into k-dimensions by using tuples of k colors per node. k-WL works on many pairs of graphs though still fails on certain cases. However, Cai et al. [2] showed the power of k-WL strictly increases with k. Interestingly, in this section, we give a connection between the expressiveness of GNNs and how well k-WL works on graph isomorphism. We let $k\text{-WL}_I$ denote the graph invariant function as the ones that reachs the stable coloring, the same as above. For equivariance, we construct a coloring of the vertices fro k-WL that is equivariant, which we denote $k\text{-WL}_E$.

Separating Power

We will use the notion of the separating power to define how well our models can discriminate between graphs. First, we provide the definition of the separating power of a set of functions, such as the sets above for LGNN, MPNN, FGNN, due to Timofte [8].

Definition 4.1. Let $\mathcal{F}$ be a set of functions f defined on set X. We define an equivalence relation $\sim$ given by $\mathcal{F}$ on X by

$$x \sim x' \iff \forall f \in \mathcal{F}, f(x) = f(x'),$$

which gives rise to the equivalence classes $\rho(\mathcal{F})$. We say that $\mathcal{F}$ is more *separating* than $\mathcal{E}$ if $\rho(\mathcal{F}) \subseteq \rho(\mathcal{E})$.

Cai et al. [2] showed that $(k+1)\text{-WL}_I$ strictly distinguishes more than $k\text{-WL}_I$ or that

$$\rho((k+1)\text{-WL}_I \subsetneq \rho(k\text{-WL}_I)$$

for $k \in \mathbb{N}$. With this notion ρ for separating power, we can summarize the current knowledge about the power of the classes of GNNs that we looked at [1].

Proposition 4.2.

$$\begin{aligned} \rho(MGNN_I) &= \rho(2\text{-}WL_I) & \rho(MGNN_E) &= \rho(2\text{-}WL_E) \\ \rho(k\text{-}LGNN_I) &= \rho(k\text{-}WL_I) & \rho(k\text{-}LGNN_E) &\subseteq \rho(k\text{-}WL_E) \\ \rho(k\text{-}FGNN_I) &= \rho((k+1)\text{-}WL_I) & \rho(k\text{-}FGNN_E) &= \rho((k+1)\text{-}WL_E) \end{aligned}$$

These results roughly give us a way to compare the discriminatory power of types of GNNs

Approximation Power

We give results for how well our models can actually approximate arbitrary invariant or equivariant functions under certain conditions. For X, Y, let $C_I(X,Y), C_E(X,Y)$ denote the set of invariant, equivariant continuous functions $X \to Y$. The closure of a family of function $\mathcal{F}$ under the uniform norm is given by $\overline{\mathcal{F}}$. Then, we can summarize the approximation power of these GNN types by the theorem below [1]

Theorem 4.3. *Let $K_{disc} \subset \mathcal{G}_0 \times \mathbb{F}_0^n, K \subseteq \mathbb{F}_0^{n^2}$ be compact. Then,*

$$\begin{aligned}
\overline{MGNN_I} &= \{f \in C_I(K_{disc}, \mathbb{F}) \mid \rho(2\text{-}WL_I) \subseteq \rho(f)\} \\
\overline{MGNN_E} &= \{f \in C_E(K_{disc}, \mathbb{F}^n) \mid \rho(2\text{-}WL_E) \subseteq \rho(f)\} \\
\overline{k\text{-}LGNN_I} &= \{f \in C_I(K, \mathbb{F}) \mid \rho(k\text{-}WL_I) \subseteq \rho(f)\} \\
\overline{k\text{-}LGNN_E} &= \{f \in C_E(K, \mathbb{F}^n) \mid \rho(k\text{-}LGNN_E \subseteq \rho(f)\} \\
&\supseteq \{f \in C_E(K, \mathbb{F}^n) \mid \rho(k\text{-}WL_E) \subseteq \rho(f)\} \\
\overline{k\text{-}FGNN_I} &= \{f \in C_I(K, \mathbb{F}) \mid \rho((k+1)\text{-}WL_I \subseteq \rho(f)\} \\
\overline{k\text{-}FGNN_E} &= \{f \in C_E(K, \mathbb{F}^n) \mid \rho((k+1)\text{-}WL_I \subseteq \rho(f)\}
\end{aligned}$$

By this theorem, we see that k-FGNNs can approximate any continuous map between a graph of order 2 (such as a normal adjacency matrix of a one) to any space $\mathbb{R}^p$ that is less separating than $(k+1)$-WL for invariant and equivariant ones.

5 Self-assessment

I would give myself a G (good) for this project. Even though I'm quite happy with how much I learned about GNNs and the end result of this report, my time management was quite bad, and I had to submite my project late.

References

[1] Waïss Azizian and Marc Lelarge. *Expressive Power of Invariant and Equivariant Graph Neural Networks*. 2020. DOI: 10.48550/ARXIV.2006.15646. URL: https://arxiv.org/abs/2006.15646.

[2] J.-Y. Cai, M. Furer, and N. Immerman. "An Optimal Lower Bound on the Number of Variables for Graph Identification". In: *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*. SFCS '89. USA: IEEE Computer Society, 1989, pp. 612–617. ISBN: 0818619821. DOI: 10.1109/SFCS.1989.63543. URL: https://doi.org/10.1109/SFCS.1989.63543.

[3] Justin Gilmer et al. "Neural Message Passing for Quantum Chemistry". In: *CoRR* abs/1704.01212 (2017). arXiv: 1704.01212. URL: http://arxiv.org/abs/1704.01212.

[4] Ningyuan Teresa Huang and Soledad Villar. "A Short Tutorial on The Weisfeiler-Lehman Test And Its Variants". In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, June 2021. DOI: 10.1109/icassp39728.2021.9413523. URL: https://doi.org/10.1109%2Ficassp39728.2021.9413523.

[5] Renjie Liao, Raquel Urtasun, and Richard S. Zemel. "A PAC-Bayesian Approach to Generalization Bounds for Graph Neural Networks". In: *CoRR* abs/2012.07690 (2020). arXiv: 2012.07690. URL: https://arxiv.org/abs/2012.07690.

[6] Haggai Maron et al. *Invariant and Equivariant Graph Networks*. 2018. DOI: 10.48550/ARXIV.1812.09902. URL: https://arxiv.org/abs/1812.09902.

[7] Benjamin Sanchez-Lengeling et al. "A Gentle Introduction to Graph Neural Networks". In: *Distill* (2021). https://distill.pub/2021/gnn-intro. DOI: 10.23915/distill.00033.

[8] Vlad Timofte. "Stone-Weierstrass theorems revisited". In: *J. Approx. Theory* 136 (2005), pp. 45–59.