Simple Tricks for Improving the Expressivity of Graph Neural Networks

David Li

Note: this blog post assumes you know the basics of graph neural networks, of which there are many good references. For example https://distill.pub/2021/gnn-intro/.

Limits of Graph Neural Networks

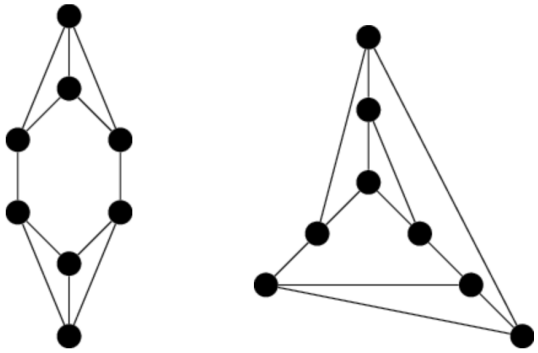
Graph neural networks (GNNs) are a powerful deep learning tool for modeling and manipulating graphs. They have been successfully applied to many fields, providing new ways to design antibiotics[1], predict protein interactions[2], enhance question-answering and recommendation systems[3][4], and understand social networks[5].

However, GNNs have their limits. By the very nature of their design, graph neural nets must be node-order equivariant. The way we encode nodes gives us information about the neighborhood of each node, but we don't capture all global information. This introduces the opportunity for ambiguity.

Two papers in 2019 (Morris et al. 2019 [6] and Xu et al. 2019 [7]) established that for the graph isomorphism problem (whether two graphs are isomorphic, e.g. you can map nodes from one graph to the other graph preserving neighbors), graph neural nets perform at most as well as the 1-dimensional Wesifeiler-Leman (WL) graph isomorphism test. The Xu et al. paper also showed that you can construct a graph neural net that achieves exactly the same performance, establishing equivalence.

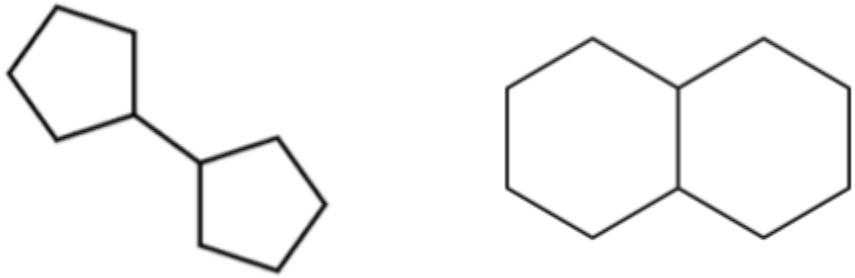
So what is the 1D WL graph isomorphism test? Well, it's pretty straightforward. We assign each node in a graph a tuple that consists of its previous label (initialized at 1) and the labels of all of its neighbors as a multiset. The label of that node is then the hash of this tuple, and we iterate hashing until convergence. If the converged hashes match in two graphs, then they are possibly isomorphic. Otherwise, they are not. Even with its simplicity, it performs well on many graphs. However, we will show a few simple graphs where this does not suffice, and thus situations where graph neural nets will not be able to distinguish.

For one, the 1D WL test and GNNs both struggle with regular graphs. For example, in the two 3-regular graphs below, the final labels for the WL test and node encodings from GNNs will be the same across both graphs. However, these graphs are not isomorphic. The one on the right has a 5-cycle, while the one on the left does not.



Source: StackExchange [8]

As an example from chemistry, graph neural nets can also struggle with non-regular graphs, for example graphs with rings. The two molecules shown below will also have the same labels in the WL test and node encodings in a GNN. Both have 10 nodes, 2 of degree 3 and 8 of degree 2. The nodes of degree 3 are each connected to a node of degree 3, and 2 nodes of degree 2. Yet, these are clearly not isomorphic. If we want to design molecules with GNNs, it might be a good idea to ensure that we can distinguish them.



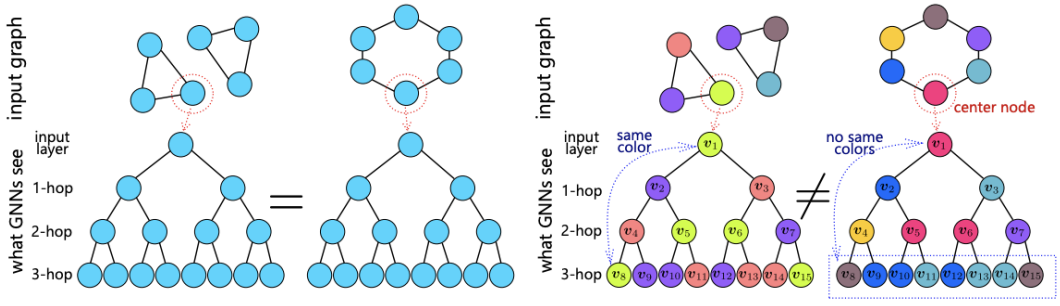
Bicyclopentyl on the left and Decalin on the right
Source: ChemDraw and Wikipedia

So, how to overcome these limits? We will show two simple tricks, providing intuition for why they work and omitting most of the technical details!

Improving the Expressivity with Random Features

One trick to improve the expressivity is to use random features at each node. Every time a procedure is called, each node is augmented with a random feature drawn from a discrete distribution of random features. This is very similar to adding unique IDs to nodes in a distributed algorithm, as they provide a way to distinguish between similar nodes.

As shown in the figure below from Sato et al. (2021) [9], adding random features to each node enhances the GNN. This figure shows the BFS tree for a given center node in each graph, which is computed during message passing. The random features enable the GNN to distinguish two graphs by comparing the original node color to the 3-hop colors and noticing that they do not match. You can also do something similar with graph colorings, like what Dasoulas et al. 2019 [10] do. However, one challenge is choosing these identifiers in a permutation equivariant way, which requires new ideas for pooling.



(a) Identical Features. (b) Random Features.

Figure 1: Illustrative example: GNNs with identical features (such as degree features) cannot distinguish a node in a cycle of three nodes with a node in a cycle of six nodes, whereas GNNs with random features can.

Figure 1 from Sato et al. (2021)

Improving the Expressivity by Counting Subgraph Structures

Another way to get over this hurdle is by encoding higher-level structures in the graph. It should be clear that if you were able to encode the entire graph somehow at each node, then this problem would be solved. However, that would defeat the entire purpose, so we have to strike an intermediate balance between just neighbors and the full graph. Augmenting in this way has the nice property of being necessarily permutation equivariant, although nodes might still remain indistinguishable, providing a trade-off between uniqueness and generalization.

Bouritsas et al., 2021 [11] maintain a set H of small connected graphs, such as cycles of fixed length or cliques. The subgraphs of the actual graph G that are isomorphic to graphs in H are then counted, and each node is augmented with a vertex structural feature that counts possible appearances of different orbits in v for each identified matching subgraph. This computation for nodes is shown on the left, with a similar computation for edges shown on the right in the figure below. This type of augmentation, and the resultant graph substructure networks (GSNs) achieve SOTA performance on a number of graph classification and regression tasks, as shown in the paper [11].

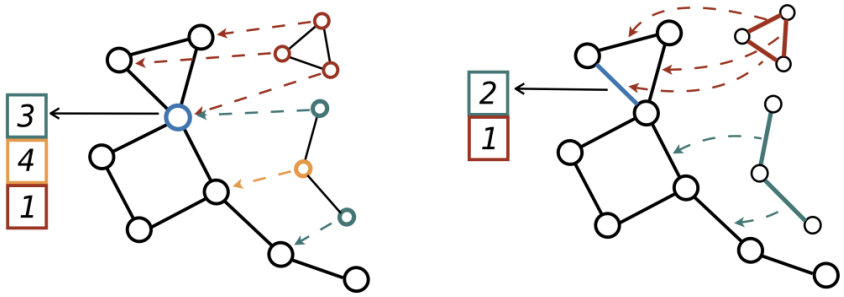


Figure 1: *Node* (left) and *edge* (right) induced subgraph counting for a 3-cycle and a 3-path. Counts are reported for the blue node on the left and for the blue edge on the right. Different colors depict orbits.

Figure 1 from Bouritsas et al (2021).

Finally, there are other methods to improve the expressivity, such as by using relational pooling [6] or higher-order GNNs.

References

1. https://www.cell.com/cell/pdf/S0092-8674(20)30102-1.pdf
2. https://arxiv.org/abs/2105.06709
3. https://arxiv.org/abs/1703.06103
4. https://arxiv.org/abs/1806.01973
5. https://www.frontiersin.org/articles/10.3389/fdata.2019.00002/full
6. https://arxiv.org/abs/1810.02244
7. https://arxiv.org/abs/1810.00826
8. https://cs.stackexchange.com/questions/105214/isomorphisms-between-regular-graphs-of-same-degree
9. https://arxiv.org/pdf/2002.03155.pdf
10. https://arxiv.org/pdf/1912.06058.pdf
11. https://arxiv.org/pdf/2006.09252.pdf