A Comparison of Various Aggregation Schemes in the Context of Graph Neural Networks

Abstract

This project gives an outline of theoretical results that establish the superiority of summation based aggregation functions and asks the question of whether this is always the case, i.e. when are max aggregators more powerful than sum aggregators?. Randomly generated Erdos-Renyi graphs and a popular real dataset on enzyme graph classification are proposed as tests. Finally, some reasoning is provided to explain why the max aggregation scheme yields the best performance. The code for the project is available here.

1 Introduction

Graph Neural Networks (GNNs) have emerged as the standard approach in solving problems pertaining to Graph Representation Learning. The core idea is to learn a *good* representation for nodes during the training setting and use it to perform inference in the test setting. While these representations are learnt at the node level, pooling schemes may be used to extend these to the graph level. This framework leads to two big questions about the notion of goodness of these representations:

1. How good are such models in learning distinct representations for distinct graphs?
2. How well do such representation learning models generalise when faced with previously unseen data?

The question of learning distinct representations for distinct graphs is fundamentally related to the graph isomorphism problem. A *good* GNN should be able to learn distinct representations for $\mathbf{z}_{\mathcal{G}_1}$ and $\mathbf{z}_{\mathcal{G}_2}$ iff $\mathcal{G}_1$ and $\mathcal{G}_2$ are not isomorphic and similar representations if they are. The graph isomorphism testing problem is known to be NP-indeterminate and hence, an approximate substitute, the Weisfeler-Leman (WL) algorithm (Leman and Weisfeiler, 1968) is used instead.

The fundamental structure of any GNN is given as follows:

$$\mathbf{h}_u^{(k+1)} = \mathsf{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathsf{AGGREGATE}^{(k)}\left(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\}\right)\right)$$

where $\mathsf{UPDATE}^{(k)}$ and $\mathsf{AGGREGATE}^{(k)}$ are arbitrary differentiable functions.

The representation of any node/graph must incorporate both the structural properties of a node's neighbourhood as well as its features and in practical settings this may as well lead to a balancing act. The former is characterised by the aggregation function and is the subject of this project.

The choice of the aggregation function represents the method by which a GNN encodes a neighbourhood's structural information in the learnt representation of a node. The aggregation function therefore, plays a direct role in determining the *expressive power* of a GNN where expressiveness captures the ability of a GNN to approximate functions on graphs. This is analogous to the universal approximation theorem for feedforward neural networks (Nalwade et al., 2024). Xu et al., 2018 show that GNNs can be at most as powerful as the WL test on classification benchmarks and establish that they are as powerful as the WL test as long as the aggregation operation and final graph level readout function (aggregating over all node representations in the final layer of a GNN) are injective and that features come from countable sets (discrete feature space). Moreover, at the graph level, they show that the sum aggregator as a readout function is inherently more expressive than the mean and max aggregators. Since the functions required to be injective are over multisets consisting of node neighbourhoods it can be shown that for any set $\mathcal{X}$, mappings of the type $f_1(\sum_{x\in\mathcal{X}} f_2(x))$ are injective as long as f_1 and f_2 are injective (Zaheer et al., 2017). Since MLPs can also represent injective functions, in principle, aggregation functions consisting of MLPs over sums are well poised to yield GNNs that match the expressive power of the WL algorithm.

Lastly, it is also possible to combine various aggregators and Corso et al., 2020 show that the sum aggregator can be generalised as a weighted combination of higher degree moments (a generalisation of the mean). Moreover, in the case where the features do not come from countable sets, a combination of n aggregators are required to be able to distinguish between multisets whose features lie in $\mathbb{R}^n$.

The subject of this project, however, is to show that there are scenarios where the sum aggregator, or any aggregator that is injective, may not perhaps be the best aggregator. As graphs get larger, more message passing layers are required to aggregate messages over larger regions leading to information bottlenecks (Alon and Yahav, 2021) which render more information useless. Sparser information (such as that coming from the max operator) may be more useful than that coming from summing, which might add up a lot of large values so that after a few layers, there is no sense of scale in the representations to be able to discern them effectively. Theoretically as well, it can be shown that there are settings where the sum aggregator does not subsume other ones (Rosenbluth et al., 2023). This problem is explored in more detail in this work.

2 Method

There are 3 experiments in this work. All three involve custom implemented message passing layers that have been reduced to the canonical formulation of the message passing layer with an element-wise ReLU function applied to each message, one of {max, mean, sum, stddev} chosen as the aggregation function (the same function is also used as the global pooling function for performing graph level classification). The objective is not to find a benchmarking setting aggregation scheme, rather it is to perform a comparative analysis and try to rationalise it. The final task is graph classification.

1. Classification Power on Random Graphs with Random Features and Structural

   Dataset: 100 Erdos-Renyi Graphs consisting of 50 nodes. The edge probabilities are $[0.1, 0.3, 0.5, 0.7]$. Node features are random vectors on $\mathbb{R}^{16}$ generated from the standard normal distribution in the first sub-experiment and represent the degree of nodes in the second case and edge probabilities serve as the classes to be learnt.

   Hyperparameters

(a) Optimiser = Adam

(b) Learning Rate = 0.001

(c) Hidden Dimensions = 16

(d) Number of Layers = 3

(e) Epochs = 150

Objective: Classify each graph correctly.

2. Classification Power as a function of the Number of Nodes in Random Graphs

   Dataset: The graphs are generated from the same distribution as before with number of nodes equal to $[10, 100, 500]$.

   Hyperparameters Same as before.

   Objective: Compute the accuracy of various aggregation schema on each of the three node cardinalities.

3. Classification Power on the ENZYMES Dataset (Morris et al., 2020)

   Dataset: The dataset consists of 600 graphs with 32.63 nodes and 62.14 edges per graph on average. The number of classes are 6 and each node has 3 features.

   Hyperparameters:

   (a) Optimiser = Adam

   (b) Learning Rate = 0.001

   (c) Hidden Dimensions = 16

   (d) Number of Layers = 3

   (e) Epochs = 300

   Objective: Compare the performance of the previously defined aggregation schemes on a real world graph dataset.

3 Results

Experiment 1

The training and testing accuracies are given in Figures 1 and 2.

In both scenarios, the max aggregator outperforms others by a significant margin. Not only is it able to achieve almost full accuracy on the training data set but its accuracy on the test set is also almost 20 percent higher than the closest one (stddev). This probably has to do with the fact that since the features are not correlated with the structure, at a local neighbourhood level, the max operator does not mix node representations, allowing for the global pooling (max) layer to aggregate information for all nodes and thus for the prediction layer to have access to the most amount of information. The weak performance of the mean aggregator is also explainable. Since features are randomly chosen from a uniform, zero-centred normal distribution, their expected value is also 0. Mean operations tend to gain no information at all about both the features as well as the structure since the averaging operation takes away any notion of magnitude of the neighbourhood messages.

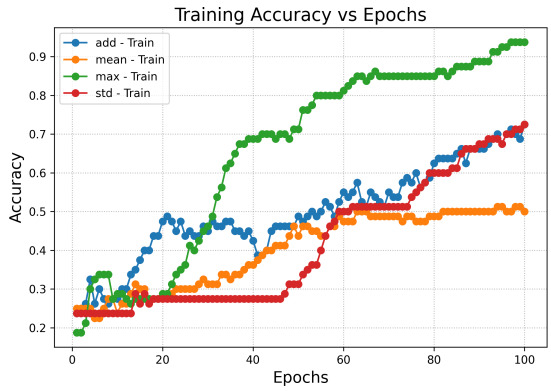

Figure 1: Training Accuracies with Random Graphs and Random Features

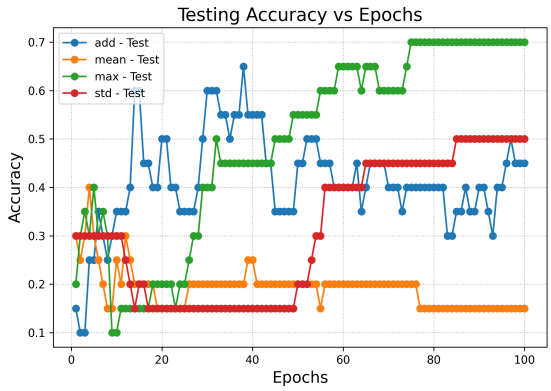

Figure 2: Testing Accuracies with Random Graphs and Random Features

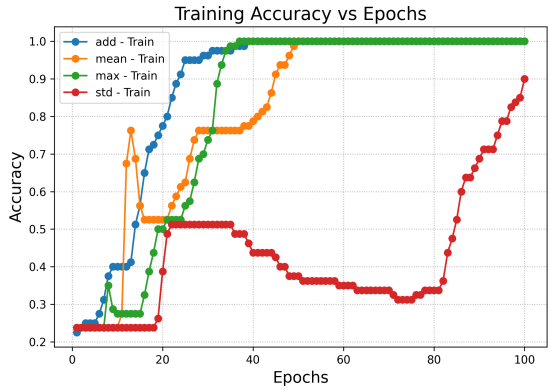

Figure 3: Training Accuracies with Random Graphs and Degree Features

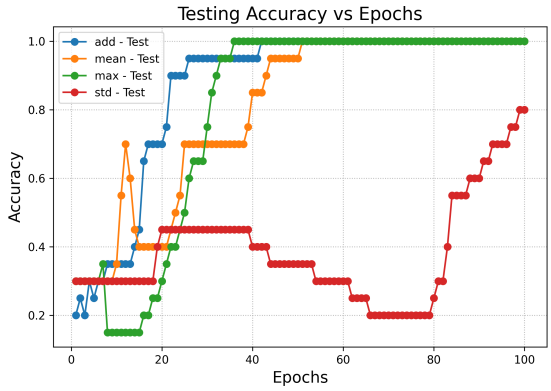

Figure 4: Testing Accuracies with Random Graphs and Degree Features

For the second sub-experiment the results are as follows:

In this case, since the features explicitly hardcode structural information coming from a countable set, an MLP over the sum of features is guaranteed to be maximally discerning. The results demonstrate its clear superiority in achieving a higher test accuracy in much fewer epochs than other schemes. Also note that, in this case, the mean aggregator also reaches optimal accuracy.

Experiment 2

The objective of this experiment was to see the variation in model accuracies upon increasing the number of nodes in the generated graph. The results, given in Table 1, show that on smaller graphs (10 nodes), summing proved to give better test accuracies, however, as the graphs became larger, the performance of the max aggregator increased across the board. This stands in line with the theoretical motivations provided in the previous part.

Aggregation/Pooling	# Nodes	Train Accuracy	Test Accuracy
Add	10	0.900	**0.550**
	100	0.625	0.500
	500	0.575	0.400
Mean	10	0.725	0.250
	100	0.575	0.200
	500	0.450	0.250
Max	10	**1.000**	0.350
	100	**1.000**	0.550
	500	**0.950**	**0.550**
Std	10	0.7875	0.200
	100	0.7750	**0.750**
	500	0.5125	0.200

Table 1: Accuracy Metrics for Different Scenarios

Experiment 3

The same aggregation schemes were tested upon the ENZYMES (Morris et al., 2020) dataset. Figures 5 and 6 show how the training and testing accuracies evolved during the training phase:

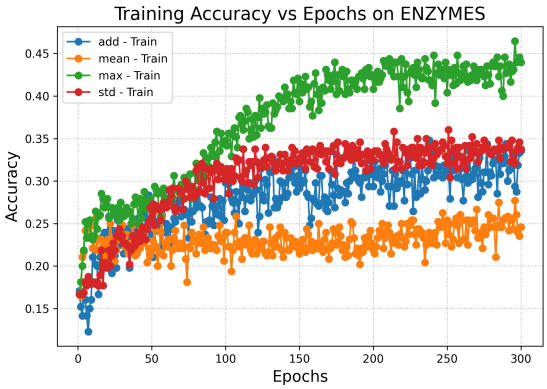

Figure 5: Training Accuracies on ENZYMES

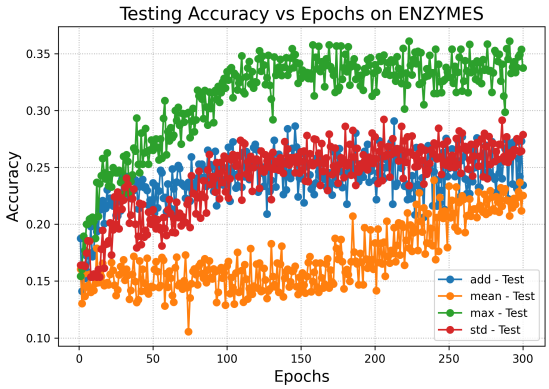

Figure 6: Testing Accuracies on ENZYMES

4 Conclusions

The experiments in this project demonstrate a significantly superior performance of the max aggregator in message passing schemes for graph classification purposes. Even though recent academic literature has shown the power of summing in expressing graph functions, we note that in cases where the graphs are larger and information bottlenecks may erode the expressive power of summations or when features may not necessarily be strongly associated with structural information (a practical scenario for many graphs where features are limited e.g. social connection graphs based on features which do not necessarily capture relational basis). The experiments proposed in this study show that indeed in such a setting, we recover a superior performance from the max aggregator. Moreover, for the same reason, we consistently find the mean aggregator as the worst performer because in such scenarios where it is important to distinguish nodes, the mean function "washes out" local information by averaging. In a sense, this work demonstrates scenarios where the standard expressive power hierarchy (sum $>$ mean $>$ max) is flipped.

References

Alon, U., & Yahav, E. (2021). On the bottleneck of graph neural networks and its practical implications. https://arxiv.org/abs/2006.05205

Corso, G., Cavalleri, L., Beaini, D., Liò, P., & Veličković, P. (2020). Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, *33*, 13260–13271.

Leman, A., & Weisfeiler, B. (1968). A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, *2*(9), 12–16.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., & Neumann, M. (2020). Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*.

Nalwade, A., Marshall, K., Eladi, A., & Sharma, U. (2024). On the expressive power of graph neural networks. https://arxiv.org/abs/2401.01626

Rosenbluth, E., Toenshoff, J., & Grohe, M. (2023). Some might say all you need is sum. https://arxiv.org/abs/2302.11603

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., & Smola, A. J. (2017). Deep sets. *Advances in neural information processing systems*, *30*.