Physics-inspired Aggregators for Graph Neural Networks - Project Proposal

Harry Shaw

November 2022

1 Introduction

1.1 Neural Networks

Multi-layer Perceptrons (MLPs) have been known theoretically to be powerful for many years [Hornik et al., 1989], though in the past 2 decades there has been an explosion in the real-world use of Neural Networks in many domains which take in rigidly structured data (i.e. vectors).

An MLP is fully-connected (figure 1) meaning every node in layer l is connected to every node in layer $l+1$ and layer $l-1$. Each layers feed-forward the vector input by carrying out an affine transformation (linear transformation $\mathbf{W}$ with bias $\mathbf{b}$), a non-linear activation function, σ, is the required as the product of many affine transformations is itself a single affine transformation. The components of $\mathbf{W}$ and $\mathbf{b}$ are the parameters of the networks that are to be learnt.

$$\mathbf{a}^{(l)} = \sigma(\mathbf{W}^{(l-1)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l-1)}) \tag{1}$$

To learn parameters, a loss function of all the parameters, J, must be defined that describes how bad the MLP currently is at carrying out its desired task. A common loss function is the sum of the squares between the network output and the desired output of some training data. MLPs often have great width and many layers, meaning there are too many parameters for J to be minimised analytically. Therefore a technique called gradient descent is required where the parameters of the neural network are nudged a little bit (nudge size $\sim \eta$) in the direction that the loss decreases the most (i.e. the gradient, J) - this is shown mathematically in equation 2 for network parameters $\mathbf{v}$.

$$\mathbf{v} \to \mathbf{v}' = \mathbf{v} - \eta \nabla J \tag{2}$$

Calculating ∇J is a non-trivial problem as altering a parameters in an early layers will alter the input of subsequent layers therefore altering the optimum parameters in these later layers. ∇J is calculated due to an algorithm called back-propagation [Rumelhart et al., 1986].

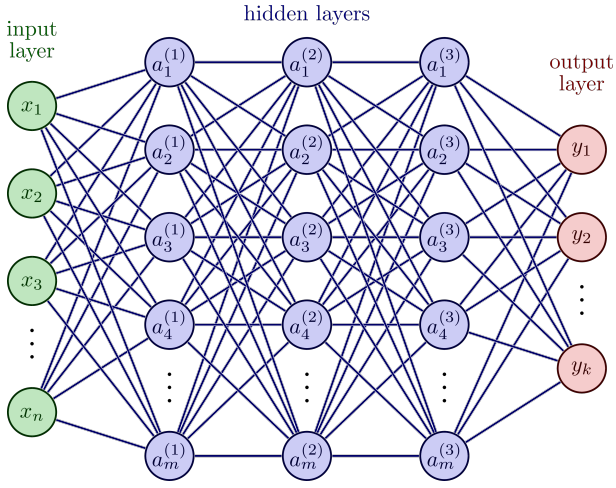

Figure 1: A Multi-layer Perceptron

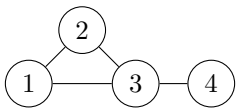

Figure 2: A simple four node graph which is represented in equation 3

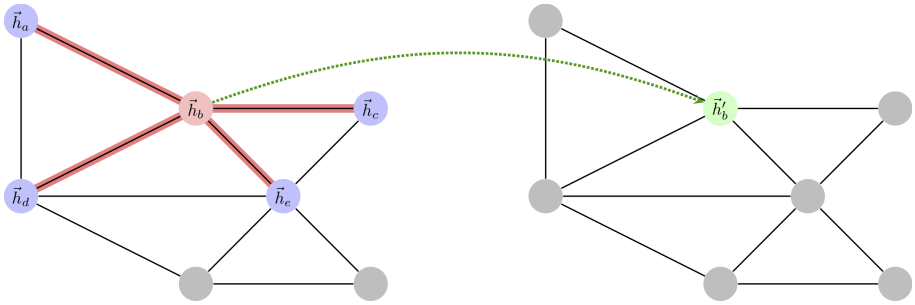

Figure 3: Message passing in GNNs

1.2 Graph Neural Networks

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$. The v-th node can be represented by a d-dimensional feature vector, $\mathbf{x}_v$. We can then construct an adjacency matrix $\mathbf{A}$ with size $|\mathcal{V}| \times |\mathcal{V}|$, with elements $A_{ij} = 1$ if $e_{ij} \in \mathcal{E}$ otherwise it is 0. Defining the neighbourhood, $\mathcal{N}$, of a node as the set of nodes that it is directly connected (i.e. 1-hop away from) to the degree matrix is given by $\mathbf{D} = \text{diag}(|\mathcal{N}_1|, ..., |\mathcal{N}_n|)$. Finally defining the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$, a matrix that will become useful later. Equation 3 shows how the graph Laplacian in calculated for the graph in figure 2, where the row number corresponds to the number of the node.

$$\underbrace{\begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}}_{\mathbf{L}} = \underbrace{\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{D}} - \underbrace{\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\mathbf{A}} \tag{3}$$

Graphs can be mapped onto many intuitive real-world situations. For example, take a social network where nodes are people and edges are friendships, each node may also have features e.g. gender, pages they follow, age etc. This node-edge relation is clearly free from any rigid structure as for example, people have different numbers of friends, whereas vectors always have the same Euclidean topology. Although it is possible to use the feature vectors in an MLP and get some meaningful results much of the rich information encoded in the connections of the graph would be lost i.e. I am more likely follow a page if many of my friends have. Therefore GNNs are certainly worth building.

$$\mathbf{x}_v^{(l)} = \gamma^{(l)} \left(\mathbf{x}_v^{(l-1)}, \oplus_{u \in \mathcal{N}(v)} \phi^{(l)} \left(\mathbf{x}_v^{(l-1)}, \mathbf{x}_u^{(l-1)} \right) \right) \tag{4}$$

Although classical algorithms have been used on graphs for over half a century (such as Djisktra's) the introduction of Graph Neural Networks (GNNs), is more recent [Scarselli et al., 2008]. Like MLPs, GNNs learn via gradient descent. Though their feed-forward operation is slightly different. Each layer of a GNN leaves the graph topology unchanged (unlike MLPs where layer width can change) and updates the feature vectors of the nodes through equation 4. The rules are constructed such that only the nodes in the neighbourhood of node v, i.e. $u \in \mathcal{N}(v)$, influence the feature vector update (see figure 3). This scheme is called message-passing. ϕ creates messages to be passed across each edge in the neighbourhood of v. These messages are then aggregated with some permutation invariant function (such as sum or max), $\oplus$, into a single message. This aggregated message is then used as an input alongside $\mathbf{x}_v^{(l-1)}$ for the function γ which calculates the new feature vector $\mathbf{x}_v^{(l)}$.

After one/many layers, GNNs can use the modified feature vectors for inference. Carrying on the social network analogy: does person X follow page Y (node classification), Are persons X and Y likely to be friends (edge prediction) and what is the purpose of a group e.g. sports team, family etc. (graph classification). For graph classification, the latent vectors are usually aggregated by a permutation invariant readout function and then passed through an MLP.

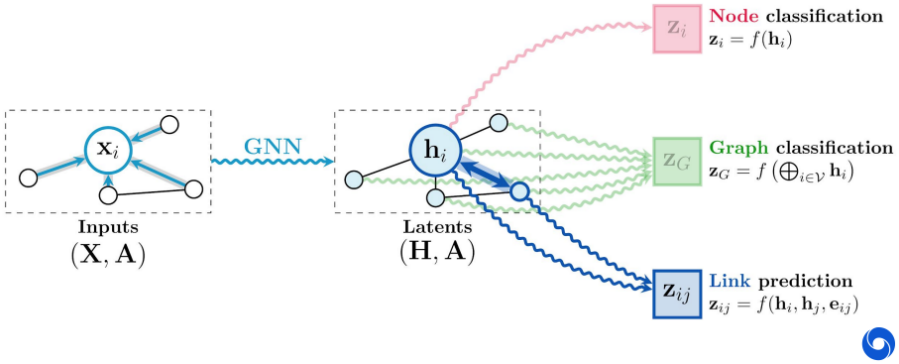

Figure 4: The three main ways a GNN can be used at inference

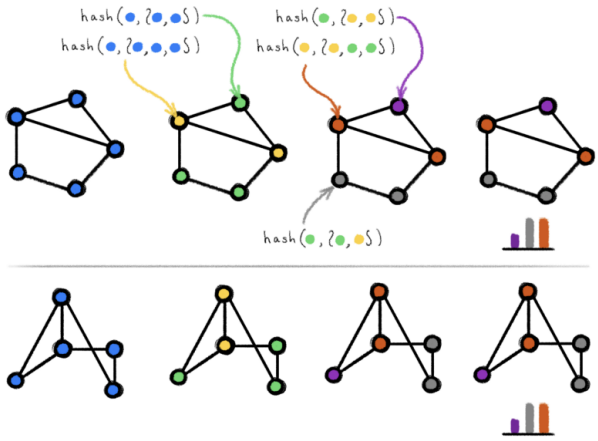

Figure 5: The Weisfeiler-Lehman test works by surveying the colours around it until the colour ratios is constant

1.3 Graph Isomorphism

It is often very hard, by eye, to see if two graphs are isomorphic (i.e. identical topology). Figure 5 shows the Weisfeiler-Lehman (WL) test [Weisfeiler and Leman, 1968]. At each stage every node receives a message from its neighbours about what colour it is, e.g. in step one, there are two blue nodes connected to three blues (and they go to yellow) and three blue nodes connected to two blues (and they go to green). This process is repeated until the ratios of colours stay constant. It should be noted that this is a process reminiscent of message-passing, features (colours) are updated according to the combination of messages received from their neighbours. Figure 5 shows two isomorphic graphs ending up with the same colour ratios, as expected.

Figure 6 shows a counterexample where non-isomorphic graphs finish with the same colours in the same ratios - it turns out that the WL test is necessary but not sufficient for isomorphism. It has been shown GNNs can be at most as good as the WL test for graph isomorphism [Xu et al., 2018]. This makes isomorphism testing an interesting property to augment in GNNs as it is a testable upper bound.

2 Augmenting $\bigoplus$

We now look at some recent work that has been done to use multiple aggregators during message passing.

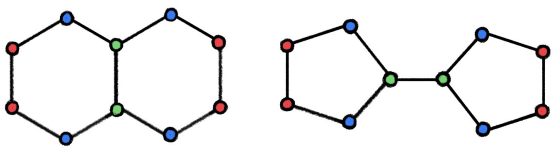

Figure 6: An example where non-isomorphic graphs have the same colours in the same ratios

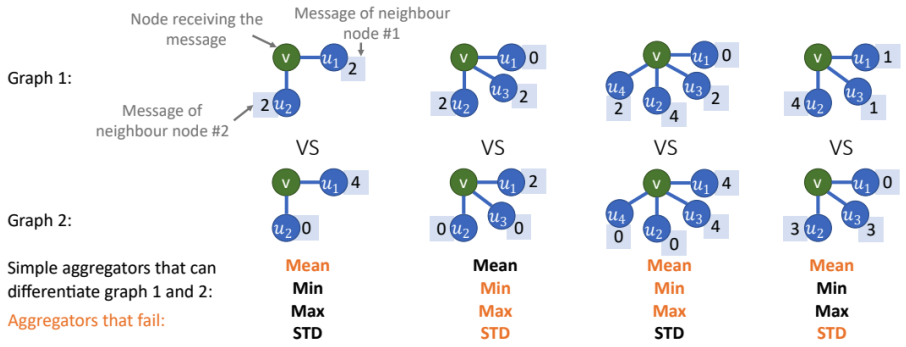

Figure 7: Different neighbourhoods that make different aggregators fail

2.1 Multiple Aggregators

We saw in the explanation of equation 4 that $\oplus$ represented a single permutation invariant aggregator (such as sum or max) in the vanilla GNN architecture. This raises the question of which aggregator is best. Figure 7 shows different neighbourhoods in which node v is receiving multiple messages from nodes u_i. For each neighbourhood four different aggregators are compared (mean, min, max and standard deviation). There is not a single aggregator that can tell apart all of the neighbourhoods suggesting that there is no single best aggregator. There is also not a single neighbourhood where all of the aggregators fail suggesting that using multiple aggregators in parallel could serve as a way to become better than the WL test as detecting isomorphism.

Principle Neighbourhood Aggregation (PNA) [Corso et al., 2020] proves Theorem 1 and Preposition 1 (for continuous features) giving very useful theoretical insight. Theorem 1 tells us that for neighbourhood $|\mathcal{N}| = n$ we need at least n aggregators to never be fooled. And Preposition 1 tells us that the moments, M_n (equation 5 happen to be a set of suitable aggregators.

Theorem 1 *(Number of aggregators needed). In order to discriminate between multisets of size n whose underlying set is $\mathbb{R}$, at least n aggregators are needed.*

Preposition 1 *(Moments of the multiset). The moments of a multiset exhibit a valid example using n aggregators*

$$M_n(X) = \sqrt[n]{\mathbb{E}[(X-\mu)^n} \quad \text{for} \quad n > 1 \in \mathbb{N} \tag{5}$$

Although theoretically suitable, higher-order moments were found to be very unstable so were not used. A practical version of the architecture was used, in which only the mean, max, min and standard deviation aggregators were used. Equation 8 gives us a new expression for $\oplus$, the tensor product means there are now 12 elements. The degree scalars are to help aggregators deal with scale better by using logarithms but are not important to our discussion.

$$\bigoplus = \begin{pmatrix} I \\ S(D, \alpha = 1) \\ S(D, \alpha = -1) \end{pmatrix} \otimes \begin{pmatrix} \mu \\ \sigma \\ \max \\ \text{mean} \end{pmatrix} \tag{6}$$

Figure 8 shows the PNA architecture, as GNN layer preserves the topology of the graph an MLP is used to turn the 12 values for the message into one via an MLP.

2.2 Directional Graph Networks

After PNA suggested multiple aggregators, people started to think - do these aggregators have to be as simple and $\mu, \sigma, \max, \min$. Usually for GNNs aggregators are isotropic (i.e. no preferred direction) but in Directional Graph Networks (DGN) [Beaini et al., 2021], the multiple aggregators are anisotropic. In graphs creating anisotropies is difficult as the topology in each neighbourhood is likely to be different. DGN suggested that meaningful global directions can be found through the eigenvectors of the graph Laplacian, $\mathbf{L}$.

To build an intuition for this, look at the time-independent Schrödinger (equation 7) for an electron in a 1-dimensional infinite square well, where $V(x) = 0$ for $0 < x < a$ and ∞ otherwise. $\psi_n(x)$ vanishes out of the well as the electron cannot have infinite energy, therefore we just need to solve the equation in the well where

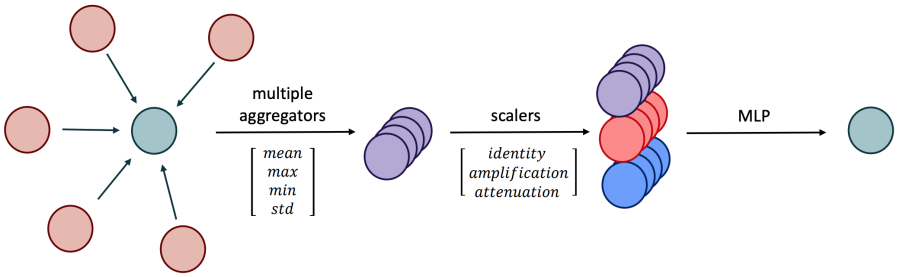

Figure 8: Architecture for PNA

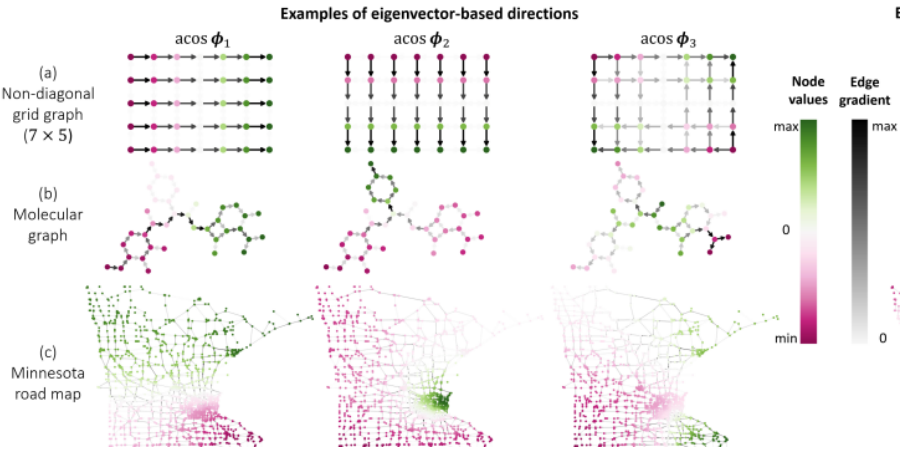

Figure 9: The projection of the eigenvectors corresponding to the three lowest non-zero eigenvalues onto some different graphs

$\nabla^2\psi_n \propto \psi_n$. The eigenfunctions are given by $\psi_n(x) \propto \sin(\frac{n\pi x}{a})$ where the eigenvalues are $E_n \propto n^2$. We see that the eigenfunction frequency grows monotonically with eigenvalue.

$$H\psi_n(x) = \left[\frac{-\hbar^2}{2m}\nabla^2 + V(x)\right]\psi_n(x) = E_n\psi_n(x) \tag{7}$$

By discretising the 1-dimensional space and treating it as a graph, it is possible to form a graph Laplacian (which should be thought of as ∇^2 acting on the line graph [Bernstein, 2020]). Leaving continuity behind means we trade eigenfunctions for eigenvectors which are sinusoidal necessarily discretised. The eigenvectors have a value for each node and we project this on. Now considering two dimensions, the lowest eigenvalues have oscillations along the longest direction (as this is allows for the longest wavelength and therefore the lowest energy), the second lowest goes in the other direction. Leaving behind Euclidean domains, this approach can be used for any graph, such as molecules, where oscillations with the lowest eigenvalues corresponding to the longest axis are still seen 9.

DGN proposed using the k-lowest eigenvectors of the graph Laplacian. Each eigenvector is projected onto the graph and is used to define two aggregators. 1) The gradient aggregator, in which the message passed is modulated by the discrete gradient of the eigenvector in that direction. 2) The directional smoothing aggregator which assigns a large weight in the forward and backwards direction of the eigenvector i.e. create low resistance flows along this direction of the eigenvector. There are $2k+1$ aggregators used as it was found that using the mean aggregator too was beneficial. It is also worth noting that DGN is provably more expressive than the WL test with respect to graph isomorphism.

$$\bigoplus = \begin{pmatrix} \mu \\ \mathbf{B}^1_{\mathrm{d}x} \\ \mathbf{B}^1_{\mathrm{av}} \\ \vdots \\ \mathbf{B}^k_{\mathrm{d}x} \\ \mathbf{B}^k_{\mathrm{av}} \end{pmatrix} \tag{8}$$

3 Proposed research

- This project aims to use the rich Physics found in the solutions to Poisson's equation to find novel anisotropic aggregators on graphs
- This project could also look at this from a Geometric perspective [Bronstein et al., 2017, Bronstein et al., 2021]. Geometric Deep Learning is a fairly new field which uses symmetries to unify neural network architectures using group theory
- Another physical approach to creating anisotropic aggregators is using effective resistance [Velingker et al., 2022] between nodes. It would be interesting to see if this resistance could be generalised to impedance with a phase.

References

[Beaini et al., 2021] Beaini, D., Passaro, S., Létourneau, V., Hamilton, W., Corso, G., and Liò, P. (2021). Directional graph networks. In *International Conference on Machine Learning*, pages 748–758. PMLR.

[Bernstein, 2020] Bernstein (2020). The graph laplacian. *https://mbernste.github.io/posts/laplacian$_m$atrix/*.

[Bronstein et al., 2021] Bronstein, M. M., Bruna, J., Cohen, T., and Velickovic, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, abs/2104.13478.

[Bronstein et al., 2017] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

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

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[Scarselli et al., 2008] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

[Velingker et al., 2022] Velingker, A., Sinop, A. K., Ktena, I., Veličković, P., and Gollapudi, S. (2022). Affinity-aware graph networks.

[Weisfeiler and Leman, 1968] Weisfeiler, B. and Leman, A. (1968). The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16.

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