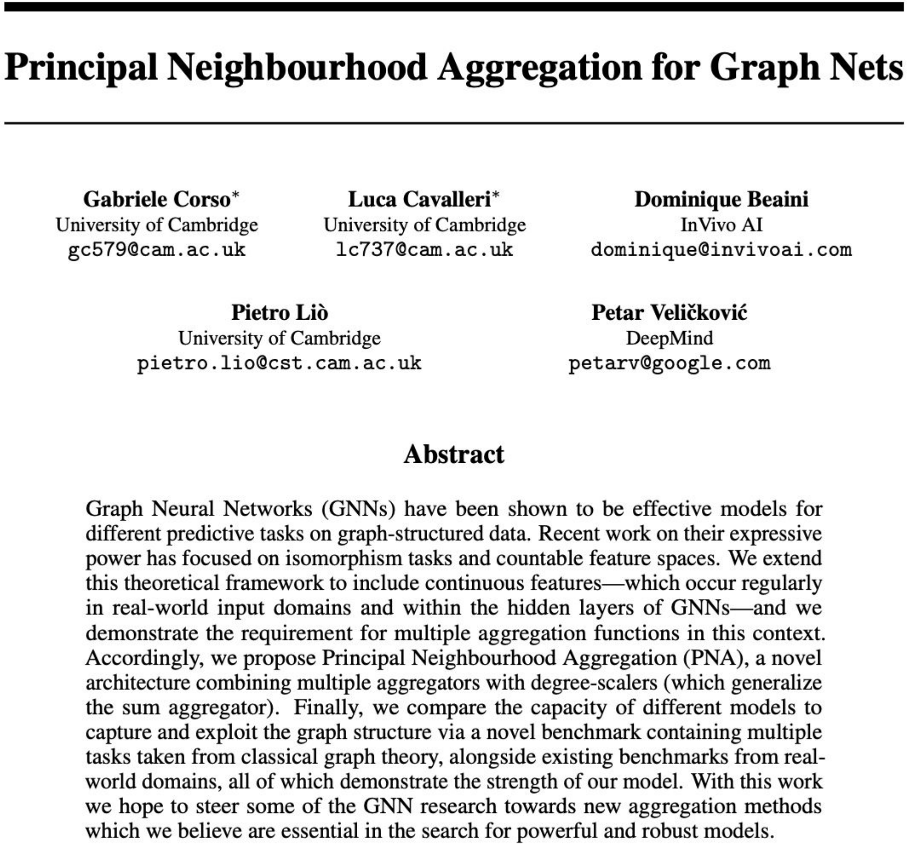
DeepMind

Principal Neighbourhood Aggregation for Graph Nets

Petar Veličković
AISC Spotlight Series
2 September 2020

What’s inside the PNA box?



Principal Neighbourhood Aggregation for Graph Nets

Gabriele Corso[*]
University of Cambridge
gc579@cam.ac.uk

Luca Cavalleri[*]
University of Cambridge
lc737@cam.ac.uk

Dominique Beaini
InVivo AI
dominique@invivoai.com

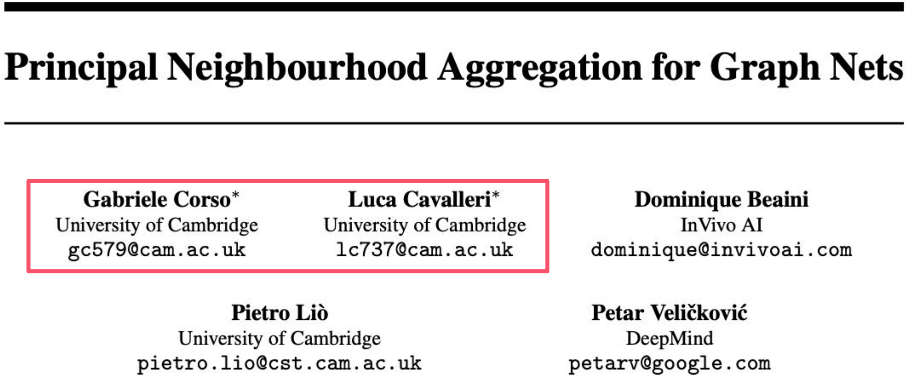
Pietro Liò
University of Cambridge
pietro.lio@cst.cam.ac.uk

Petar Veličković
DeepMind
petarv@google.com

Abstract

Graph Neural Networks (GNNs) have been shown to be effective models for different predictive tasks on graph-structured data. Recent work on their expressive power has focused on isomorphism tasks and countable feature spaces. We extend this theoretical framework to include continuous features—which occur regularly in real-world input domains and within the hidden layers of GNNs—and we demonstrate the requirement for multiple aggregation functions in this context. Accordingly, we propose Principal Neighbourhood Aggregation (PNA), a novel architecture combining multiple aggregators with degree-scalers (which generalize the sum aggregator). Finally, we compare the capacity of different models to capture and exploit the graph structure via a novel benchmark containing multiple tasks taken from classical graph theory, alongside existing benchmarks from real-world domains, all of which demonstrate the strength of our model. With this work we hope to steer some of the GNN research towards new aggregation methods which we believe are essential in the search for powerful and robust models.

What’s inside the PNA box?



Principal Neighbourhood Aggregation for Graph Nets

Gabriele Corso*
University of Cambridge
gc579@cam.ac.uk

Luca Cavalleri*
University of Cambridge
lc737@cam.ac.uk

Dominique Beaini
InVivo AI
dominique@invivoai.com

Pietro Liò
University of Cambridge
pietro.lio@cst.cam.ac.uk

Petar Veličković
DeepMind
petarv@google.com

Expressivity of GNNs
Aggregation functions

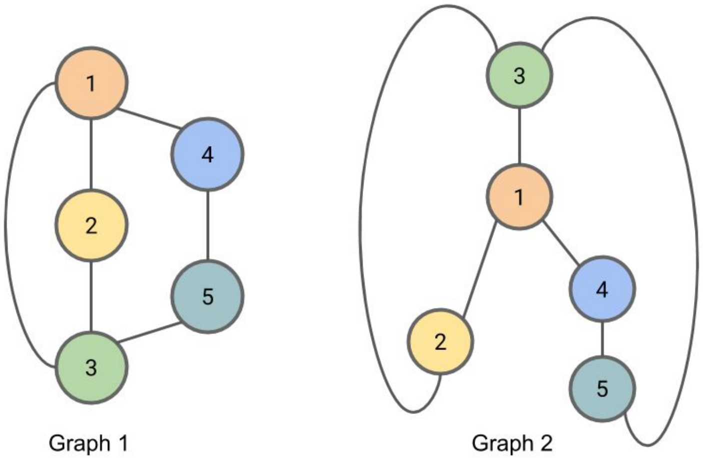
PNA Architecture

https://arxiv.org/abs/2004.05718

How *powerful* are Graph Neural Networks?

- GNNs are a powerful tool for processing real-world graph data
  - But they won't solve ***any*** task specified on a graph accurately!

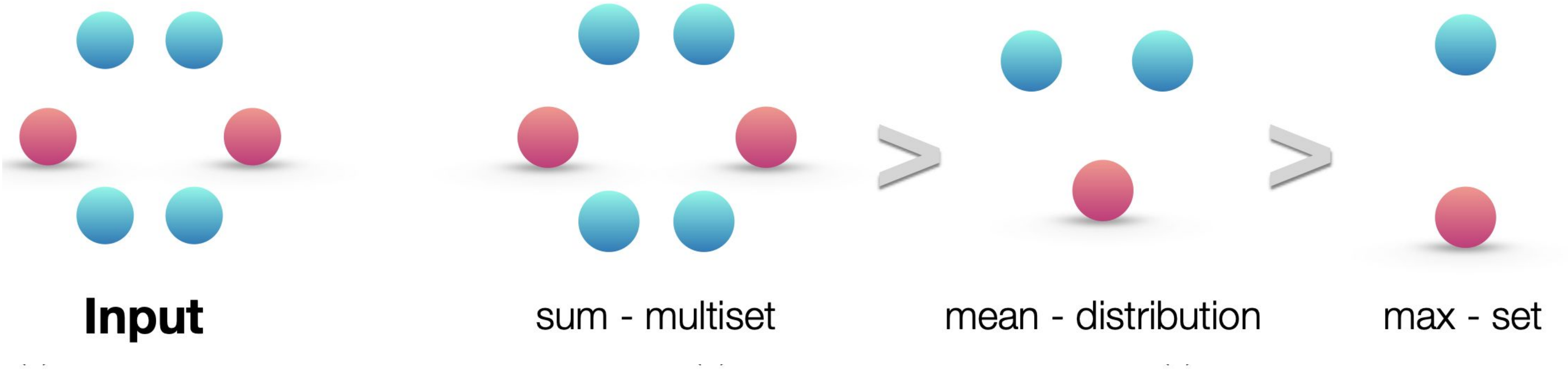
- Canonical example: deciding *graph isomorphism*
  - Am I able to use my GNN to **distinguish** two non-isomorphic graphs?
  - (Permutation invariance mandates isomorphic graphs will be detected)



Prior art (Xu *et al.*, ICLR’19)

- We can relax the problem by looking first at distinguishing **neighbourhoods**

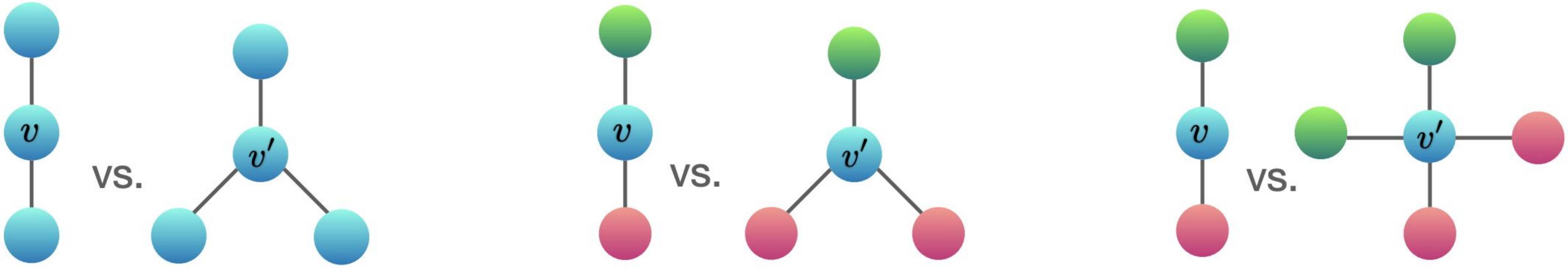
- For the case of **discrete feature spaces**, it is shown that *aggregation* function choice can vastly influence the expressive power:



Prior art (Xu *et al.*, ICLR’19)

- We can relax the problem by looking first at distinguishing **neighbourhoods**

- For the case of **discrete feature spaces**, it is shown that *aggregation* function choice can vastly influence the expressive power:



(a) Mean and Max both fail

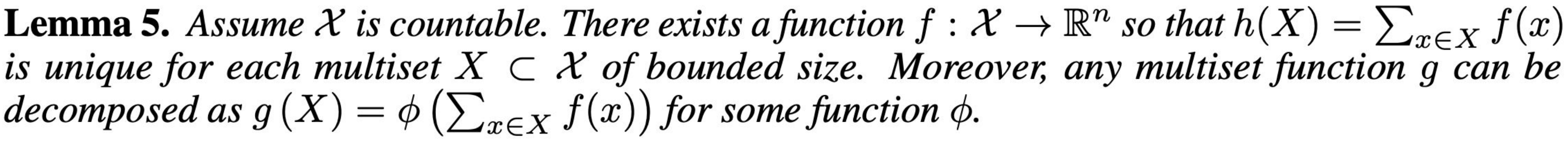
(b) Max fails

(c) Mean and Max both fail

- It appears that **sum** is a very important primitive!

Optimally expressive GNNs (Xu *et al.*, ICLR'19)

- The sum aggregation is, actually, **injective** in this context: it will never map two different neighbourhoods to the same output!



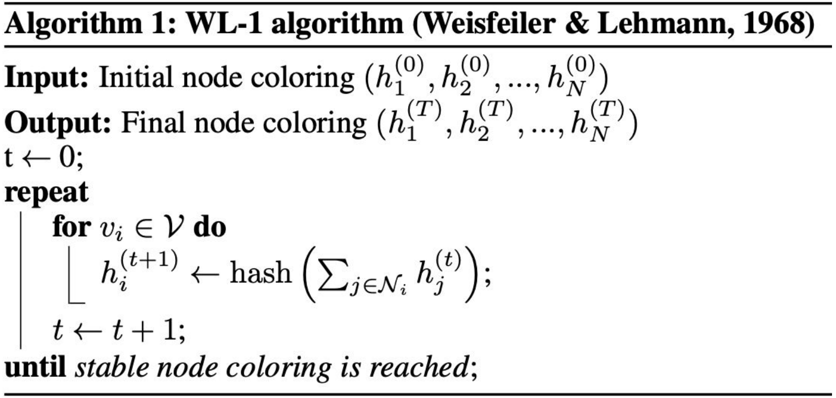
Lemma 5. *Assume $\mathcal{X}$ is countable. There exists a function $f : \mathcal{X} \to \mathbb{R}^n$ so that $h(X) = \sum_{x \in X} f(x)$ is unique for each multiset $X \subset \mathcal{X}$ of bounded size. Moreover, any multiset function g can be decomposed as $g(X) = \phi\left(\sum_{x \in X} f(x)\right)$ for some function ϕ.*

- Combining **sum** aggregation with appropriately chosen message functions yields **optimally** expressive GNNs in this setting.
    - *Graph Isomorphism Network* (**GIN**)

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

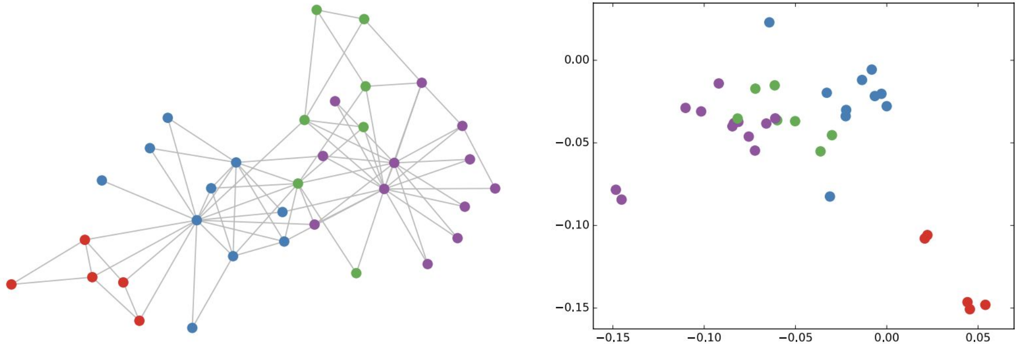
Optimally expressive GNNs (Xu *et al.*, ICLR'19)

- In fact, they showed that no spatial GNN can ever perform better than **Weisfeiler-Leman**:



Algorithm 1: WL-1 algorithm (Weisfeiler & Lehmann, 1968)

```
Input: Initial node coloring $(h_1^{(0)}, h_2^{(0)}, ..., h_N^{(0)})$
Output: Final node coloring $(h_1^{(T)}, h_2^{(T)}, ..., h_N^{(T)})$
$t \leftarrow 0$;
repeat
    for $v_i \in \mathcal{V}$ do
        $h_i^{(t+1)} \leftarrow \text{hash}\left(\sum_{j \in \mathcal{N}_i} h_j^{(t)}\right)$;
    $t \leftarrow t + 1$;
until stable node coloring is reached;
```



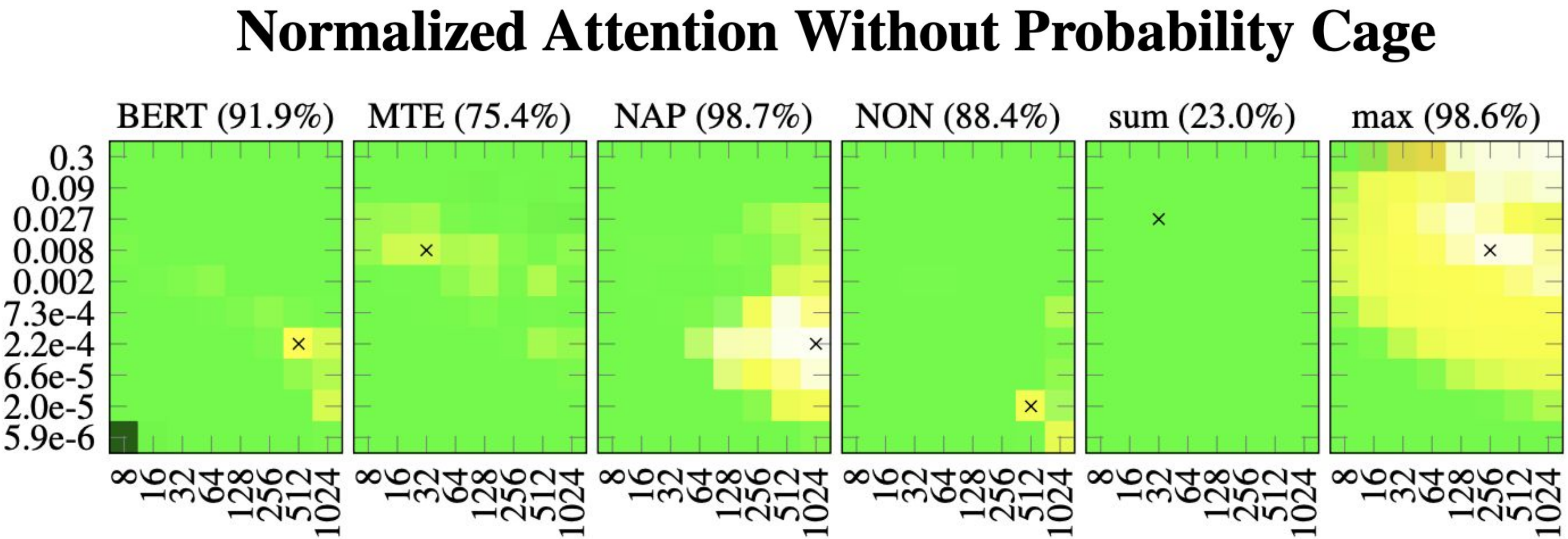
(a) Karate club network (b) Random weight embedding

- Several works try to propose works that go ***beyond*** 1-WL
    - *Relational Pooling (Murphy et al., ICML'19), 1-2-3-GNN (Morris et al., AAAI'19), ...*

Notes on Xu *et al.*

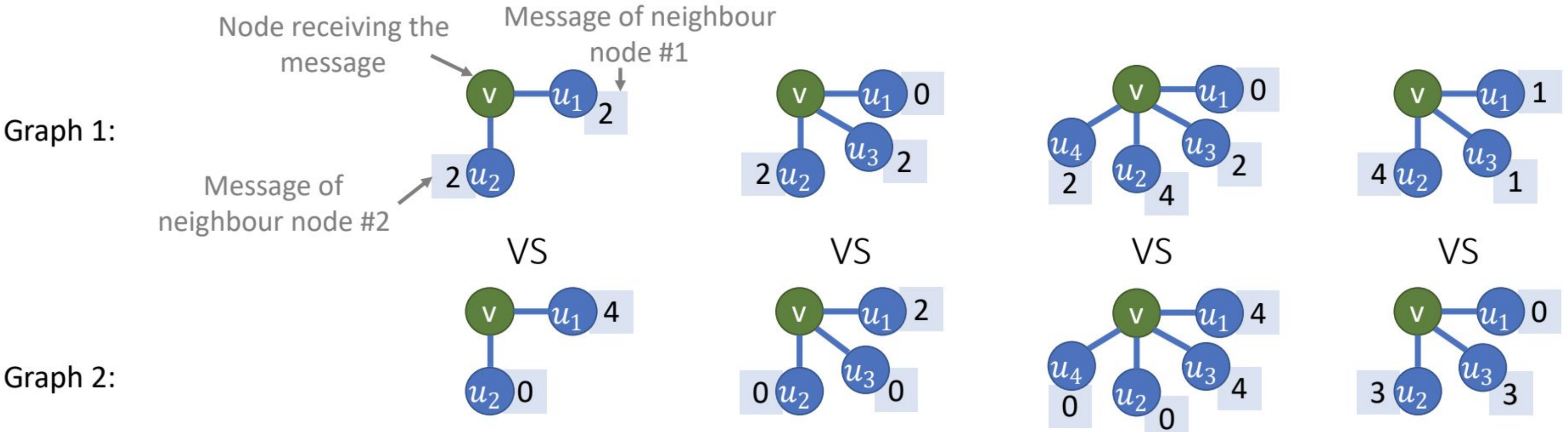
- Does this mean **max** is useless? **NO!** *Not all tasks are isomorphism.*
  - In practice, sum can cause *exploding messages*.
  - Max dominates on right kinds of problems (e.g. sparse credit assignment / search)

(Richter and Wattenhofer. 2020)



Notes on Xu *et al.*

- What happens when features are **continuous**? (real-world apps / latent GNN states)
  - … the proof for injectivity of sum (hence GINs' expressivity) **falls apart**


Node receiving the message
Message of neighbour node #1
Message of neighbour node #2
Graph 1:
Graph 2:
VS
v
u_1
u_2
u_3
u_4
2
2
0
2
2
0
2
2
4
1
4
1
4
0
2
0
0
4
0
4
0
0
0
3
3

Simple aggregators that can differentiate graph 1 and 2:	Min Max STD	Mean	STD	Min Max
Aggregators that fail:	Mean	Min Max STD	Mean Min Max	Mean STD

Which is best? **Neither.**

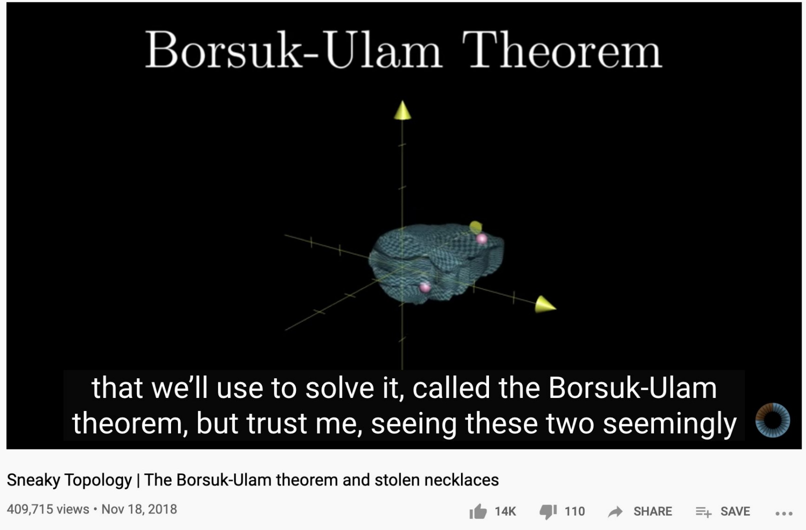
- There doesn’t seem to be a clear single “winner” aggregator here…

- In fact, we prove that **there isn’t one**!

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.*

- The proof is (in my opinion) **really cool**!

  (relies on **Borsuk–Ulam** theorem)



that we’ll use to solve it, called the Borsuk-Ulam theorem, but trust me, seeing these two seemingly

Sneaky Topology | The Borsuk-Ulam theorem and stolen necklaces

409,715 views • Nov 18, 2018

The proof

Proof. Let S be the n-dimensional subspace S of $\mathbb{R}^n$ formed by all tuples $(x_1, x_2, \ldots, x_n)$ such that $x_1 \leq x_2 \leq \ldots \leq x_n$, and notice how S is the collection of the aforementioned multisets. We defined an aggregator as a continuous function from multisets to reals, which corresponds to a continuous function $g : S \rightarrow \mathbb{R}$.

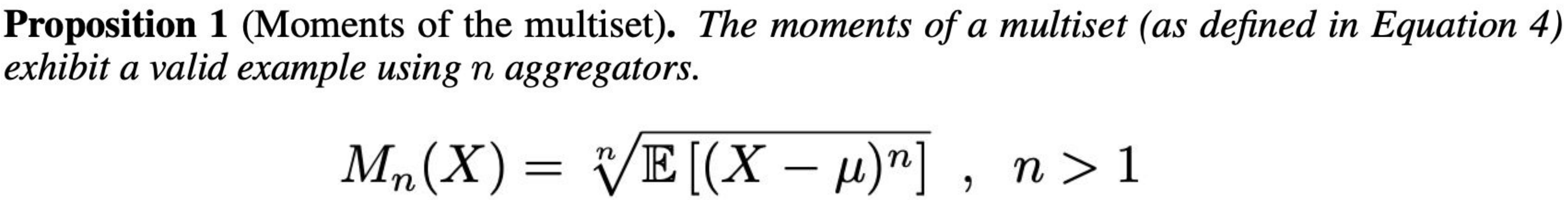
Assume by contradiction that it is possible to discriminate between all the multisets of size n using only $n - 1$ aggregators, viz. $g_1, g_2, \ldots, g_{n-1}$.

Define $f : S \rightarrow \mathbb{R}^{n-1}$ to be the function mapping each multiset X to its output vector $(g_1(X), g_2(X), \ldots, g_{n-1}(X))$. Since $g_1, g_2, \ldots, g_{n-1}$ are continuous, so is f, and, since we assumed these aggregators are able to discriminate between all the multisets, f is injective.

As S is a n-dimensional Euclidean subspace, it is possible to define a $(n-1)$-sphere C^{n-1} entirely contained within it, i.e. $C^{n-1} \subseteq S$. According to Borsuk–Ulam theorem [34, 35], there are two distinct (in particular, non-zero and antipodal) points $\vec{x}_1, \vec{x}_2 \in C^{n-1}$ satisfying $f(\vec{x}_1) = f(\vec{x}_2)$, showing f not to be injective; hence the required contradiction. □

Okay, but what *are* these *n* aggregators?

- Multiset **moments** work!



Proposition 1 (Moments of the multiset). *The moments of a multiset (as defined in Equation 4) exhibit a valid example using* n *aggregators.*

$$M_n(X) = \sqrt[n]{\mathbb{E}\left[(X-\mu)^n\right]} \ , \quad n > 1$$

- **N.B.** This covers aggregators like *mean, standard deviation, …*
  - And could explain why max works well at times:

    max and min together form an estimate of **M∞**! *(similar insights in Adamax)*

- We don't prove that moments are always the optimal choice…
  - But do prove that they satisfy the theoretical constraints for neighbour isomorphism

Moments and scalers

- Note that we’ve excluded previously useful aggregators like **sum**
  - Consider an interesting observation: **sum ~ mean** o **degree scaler**!
  - Also consider **logarithmic** and **exponential** scalers, to highlight *hubs* and *authorities*

$$S(d, \alpha) = \left(\frac{\log(d+1)}{\delta} \right)^{\alpha}, \quad d > 0, \quad -1 \leq \alpha \leq 1$$

- Adding higher-order moments could quickly lead to **numerical instability**
  - Have to take n-th powers followed by n-th roots…

Principal Neighbourhood Aggregation (PNA)

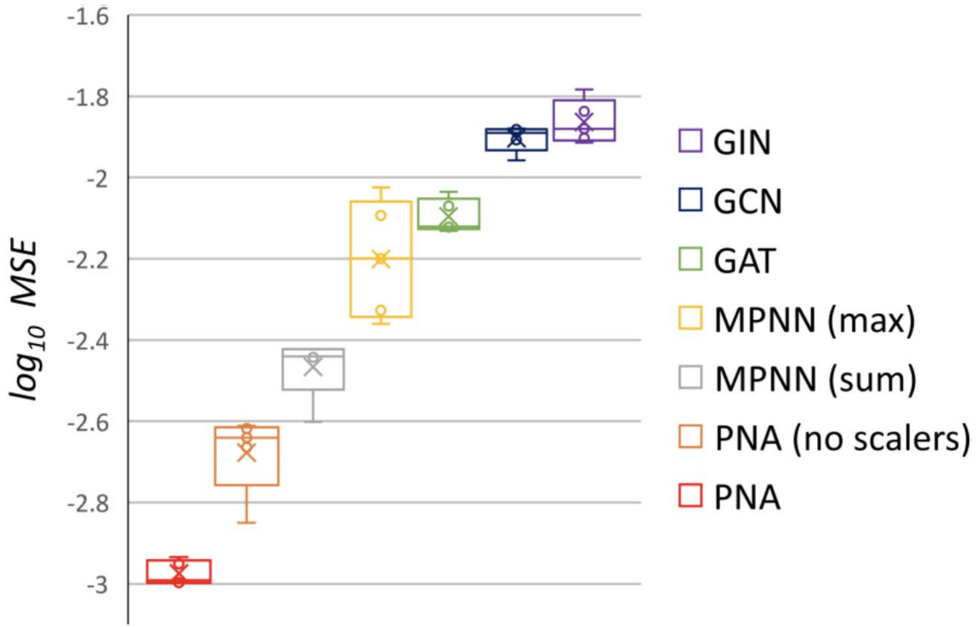
- With all of the above in mind, we propose **Principal Neighbourhood Aggregation**
  - a **robust** aggregation scheme which incorporates the necessary principles

$$\bigoplus = \underbrace{\begin{bmatrix} I \\ S(D, \alpha = 1) \\ S(D, \alpha = -1) \end{bmatrix}}_{\text{scalers}} \otimes \underbrace{\begin{bmatrix} \mu \\ \sigma \\ \max \\ \min \end{bmatrix}}_{\text{aggregators}}$$

- Stitch into your favourite GNN model and you're all set!

$$X_i^{(t+1)} = U\left(X_i^{(t)}, \bigoplus_{(j,i)\in E} M\left(X_i^{(t)}, X_j^{(t)}\right)\right)$$

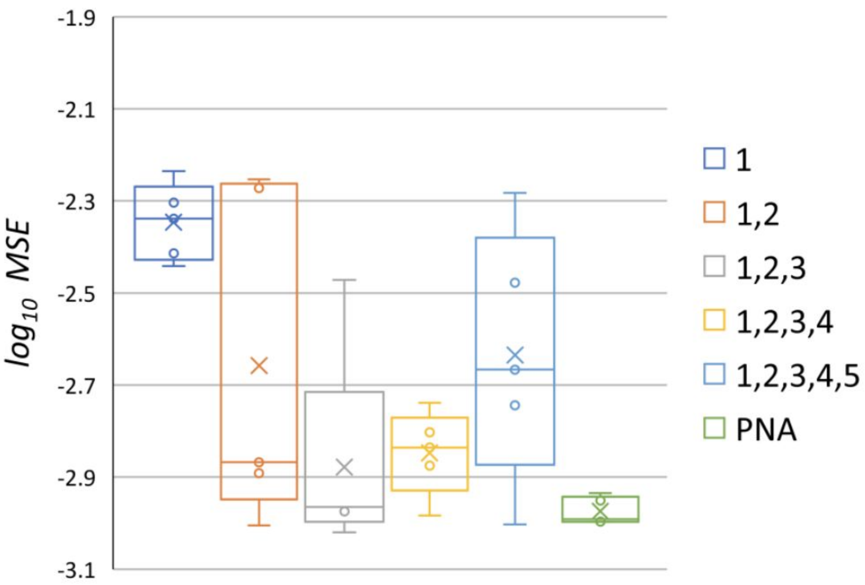
PNA on synthetic **graph property prediction**



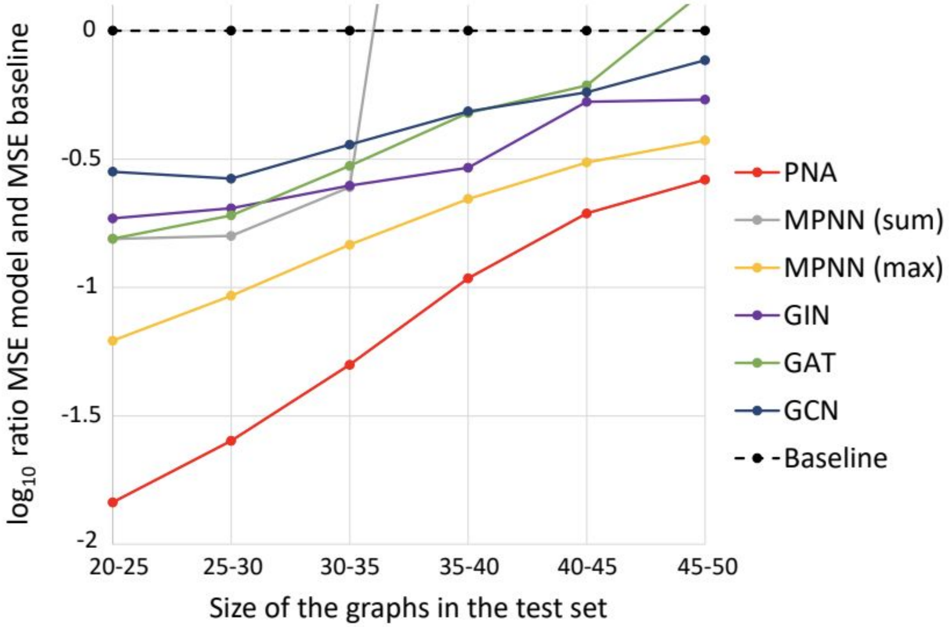
		Nodes tasks			Graph tasks		
Model	**Average score**	**1**	**2**	**3**	**4**	**5**	**6**
PNA	**-3.13**	**-2.89**	**-2.89**	**-3.77**	**-2.61**	**-3.04**	**-3.57**
PNA (no scalers)	-2.77	-2.54	-2.42	-2.94	**-2.61**	-2.82	-3.29
MPNN (sum)	-2.53	-2.36	-2.16	-2.59	-2.54	-2.67	-2.87
MPNN (max)	-2.50	-2.33	-2.26	-2.37	-1.82	-2.69	-3.52
GAT	-2.26	-2.34	-2.09	-1.60	-2.44	-2.40	-2.70
GCN	-2.04	-2.16	-1.89	-1.60	-1.69	-2.14	-2.79
GIN	-1.99	-2.00	-1.90	-1.60	-1.61	-2.17	-2.66
Baseline	-1.38	-1.87	-1.50	-1.60	-0.62	-1.30	-1.41

1. Single-source shortest-paths
2. Eccentricity
3. Laplacian features
4. Connected
5. Diameter
6. Spectral radius

Stability is important, max is relevant!

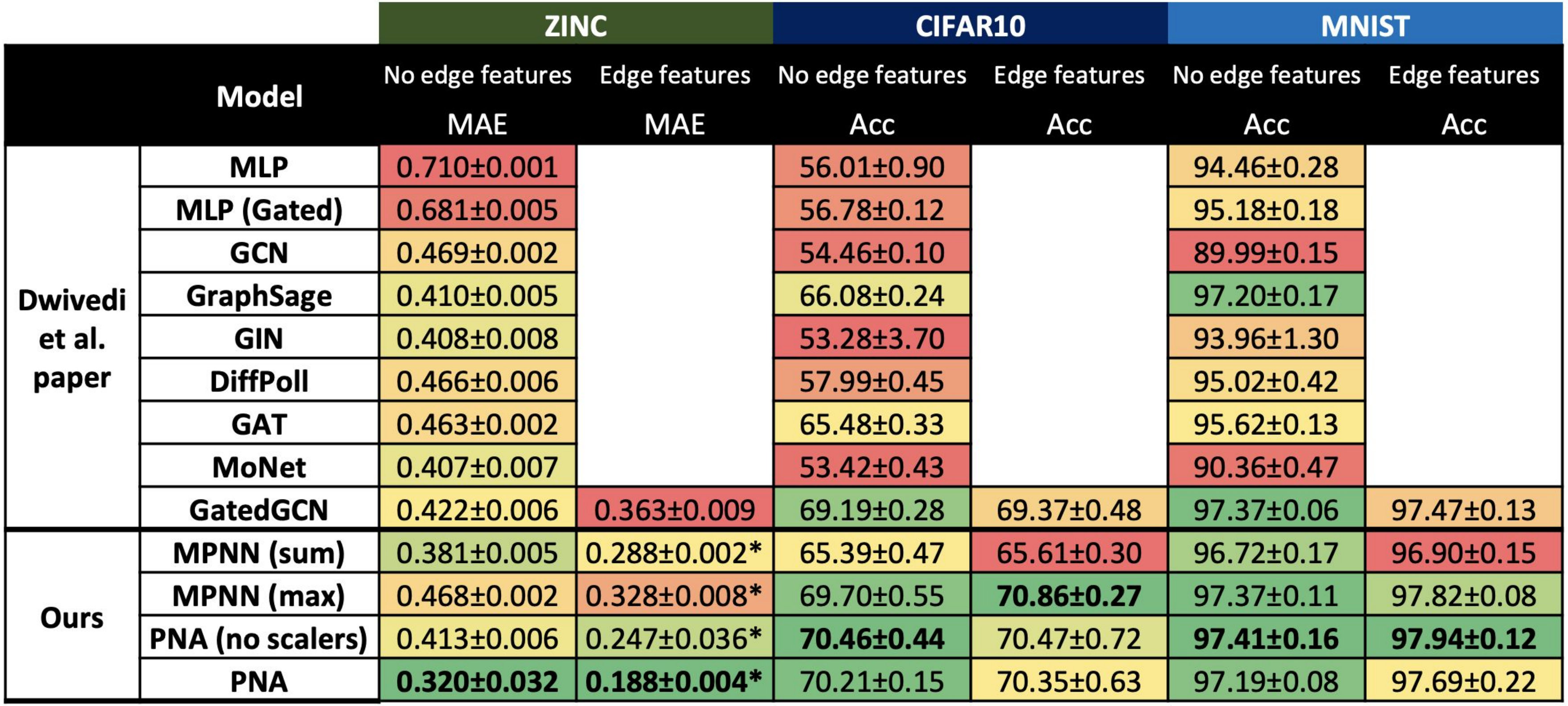


Adding more moments



Scaling up test graphs

PNA works in the real world



		ZINC		CIFAR10		MNIST	
	Model	No edge features	Edge features	No edge features	Edge features	No edge features	Edge features
		MAE	MAE	Acc	Acc	Acc	Acc
Dwivedi et al. paper	**MLP**	0.710±0.001		56.01±0.90		94.46±0.28	
	MLP (Gated)	0.681±0.005		56.78±0.12		95.18±0.18	
	GCN	0.469±0.002		54.46±0.10		89.99±0.15	
	GraphSage	0.410±0.005		66.08±0.24		97.20±0.17	
	GIN	0.408±0.008		53.28±3.70		93.96±1.30	
	DiffPoll	0.466±0.006		57.99±0.45		95.02±0.42	
	GAT	0.463±0.002		65.48±0.33		95.62±0.13	
	MoNet	0.407±0.007		53.42±0.43		90.36±0.47	
	GatedGCN	0.422±0.006	0.363±0.009	69.19±0.28	69.37±0.48	97.37±0.06	97.47±0.13
Ours	**MPNN (sum)**	0.381±0.005	0.288±0.002*	65.39±0.47	65.61±0.30	96.72±0.17	96.90±0.15
	MPNN (max)	0.468±0.002	0.328±0.008*	69.70±0.55	**70.86±0.27**	97.37±0.11	97.82±0.08
	PNA (no scalers)	0.413±0.006	0.247±0.036*	**70.46±0.44**	70.47±0.72	**97.41±0.16**	**97.94±0.12**
	PNA	**0.320±0.032**	**0.188±0.004***	70.21±0.15	70.35±0.63	97.19±0.08	97.69±0.22

Key takeaways

- One key way to measure GNN expressive power is ***neighbourhood isomorphism***
  - "Can the GNN layer distinguish neighbourhoods?"

- When features are continuous, ***multiple aggregators*** *are needed!*
  - Sum is no longer sufficient (and may often be inappropriate)!
  - The *n* moments are one example of such aggregator set

- Combine a stable subset of moments with scalers => yield the **PNA** architecture
  - Strong performance on synthetic and real-world benchmarks
  - Can "latch-on" to the strongest aggregator

- Designing strong aggregators still very much an **open area** of research!

Goodies

- Paper @ arXiv:

  https://arxiv.org/abs/2004.05718

- Code @ GitHub:

  https://github.com/lukecavabarrett/pna

- Promo video @ ICML'20 GRL+:

  https://slideslive.com/38931510/

- Implementation @ PyTorch Geometric:

  https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html#torch_geometric.nn.conv.PNAConv

DeepMind

Thank you!

Questions?

`petarv@google.com | https://petar-v.com`

In collaboration with Gabriele Corso, Luca Cavalleri, Dominique Beaini and Pietro Liò