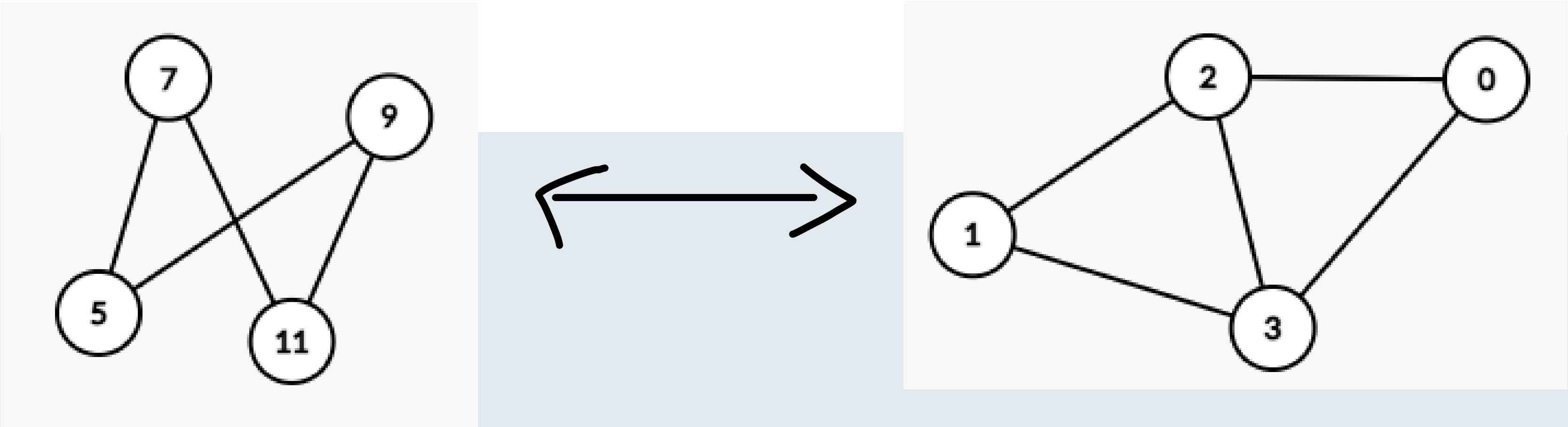
The Expressive Power of Graph Neural Networks

Taj Jones-McCormick
CS 886



Note: This reading covers sections 5.1 - 5.4 (by Pan Li and Jure Leskovec) of a textbook on GNN's, and covers results from a few papers

Introduction

- What is expressive Power?
  - Some model $F_\theta(x)$
  - What functions can we represent (approximately)
  - By choosing different parameters θ

- How to measure this?

- We want to know the expressive power of GNNs!

Background

Universal Approximation theorem (Cybenko, 1989):

- 1 hidden layer NNs (sigmoid) can approximate any continuous function on bounded interval (needs to be sufficiently wide)

- Other examples?
  - Stone-Weierstrass theorem – how to prove expressivity over C[a, b]
  - Polynomial approximation
  - Haar Wavelet
  - Fourier Analysis
  - Measurable functions from simple functions

The importance

- Underlying theory of models relies on their expressivity
  - Contributes to our understanding
- Practitioners should know what models are capable of
- When to use which method?

- Graph Neural Nets are powerful for large class of problems
- When to use them? When will they be good enough?

Why don't previous methods work?

- GNNs unique architecture's => need their own results
- What functions do we want to approximate?
  - Inductive biases influence function class
    - Ex. CNN – translation invariance
    - RNN – time invariance
    - GNN – permutation invariance (fundamental assumption)

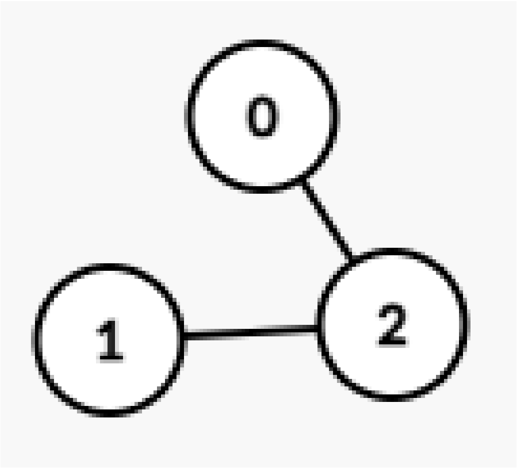
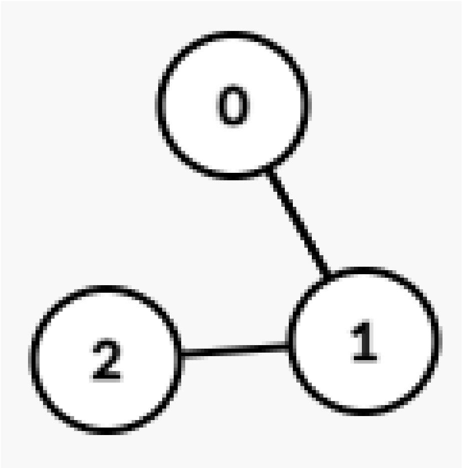
- More restrictive class of functions creates new challenges
  - Can't simply use Stone-Weierstrass type tools

Permutation Invariance and Isomorphisms

- Permutation invariance
  - Our model should not care how our graph is encoded

- Isomorphic Graphs:
  - If there exists a bijective function between nodes, preserving edges
  - 2 graphs A, B (nxn, adjacency matrices)
  - For some permutation, ie bijective map: $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$
  - We have $A_{i,j} = B_{\pi(i),\pi(j)}$

- A model is permutation invariant if it acts the same on any pair of isomorphic graphs

 $\cong$ 

$\|$ $\|$

$$\begin{pmatrix}0&0&1\\0&0&1\\1&1&0\end{pmatrix} \cong \begin{pmatrix}0&1&0\\1&0&1\\0&1&0\end{pmatrix}$$

Expressiveness for Permutation Invariant Functions?

- Zaheer et al 2017 (DeepSet) show that any function F is continuous and permutation invariant, iff it can be written in the form:

$$F(v) = \phi(\textstyle\sum_{a \in v} \rho(a))$$

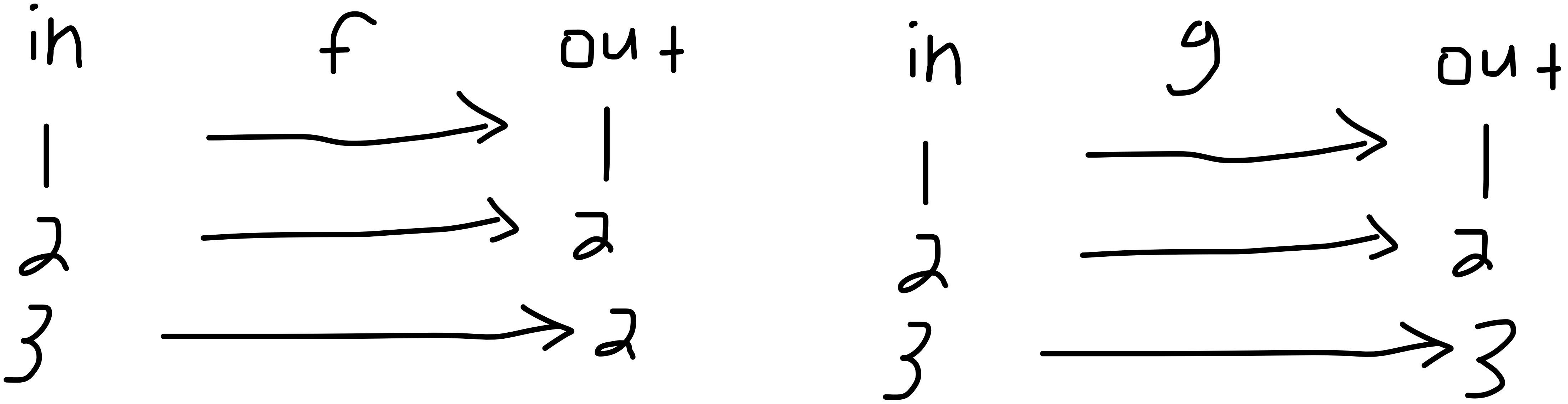
- Where ϕ and ρ are continuous functions, v countable

- This gives an easy way to construct models with strong expressiveness over permutation invariant continuous functions

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

A new definition for Expressive Power

- How well can a model differentiate between non-isomorphic graphs?
- How many inputs to a model correspond to unique outputs?



- A weaker definition than before
- Differentiation does not imply approximation but:
- No differentiation implies no approximation

Review: Message Passing GNN's (MP-GNN)

- Given some graph: $G = (V, \epsilon, X)$
- Set Node representations: $h_v^0 = X_v, v \in V$
- Recursively perform:

$$m_{u,v}^l = msg(h_u^{l-1}, h_v^{l-1})$$

$$A_v^l = aggregate(m_{v,u}^l, \forall u \in \epsilon_v)$$

$$h_v^l = update(h_v^{l-1}, A_v^l)$$

$$\hat{y} = predict(h_v^l, \forall v \in V)$$

-'msg', 'update', 'predict' are neural nets. 'aggregate' is permutation invariant function like a summation

- MP-GNNs are permutation invariant! Recall: $F(v) = \phi(\sum_{a \in v} \rho(a))$

Review: Weisfeiler-Lehman test (1d)

- How to check if graphs are isomorphic?
  - Can look at permutations of adjacency matrices O(n!), way too slow

- The WL test:
  - Representation for each node (partition nodes based on representation)
  - Hash each node's representation + set of neighbours representations
  - Arrive at new representation and partition (hashing is injective)
  - Iterate until convergence and compare partitions

  Not perfect!

Equivalence of GNN and WL-test, GIN

- Theorem:
  - If MP-GNN maps two graphs to different representations, then WL will decide they are not isomorphic
    - (upper bound MP-GNN expressivity)

- Matching the upper bound:
  - Ensure that the right components of MP-GNN are injective (like hashing in WL)
  - Use summation as invariant pooling (injective, unlike max, mean)
  - This is the Graph Isomorphic network

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

Limitations of WL-test (and thus MP-GNN)

- Cycles can cause problems also:

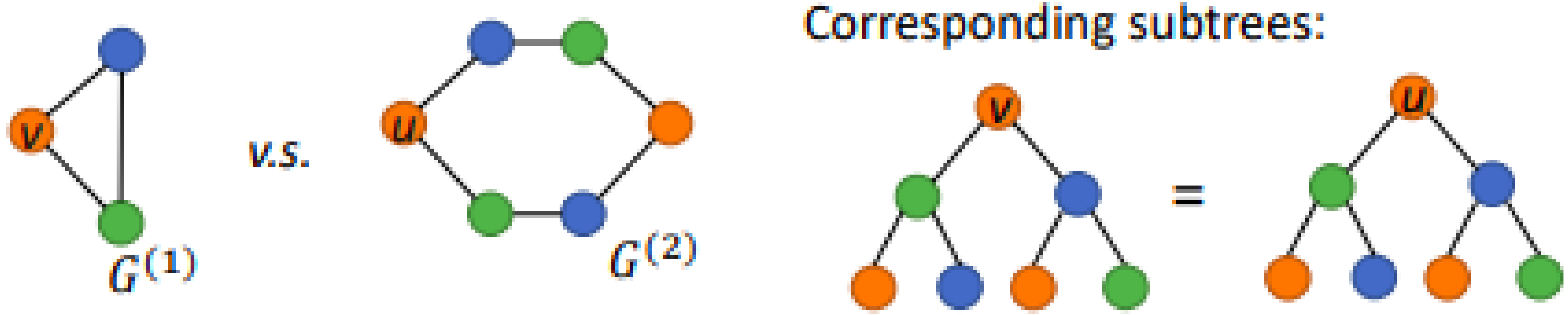
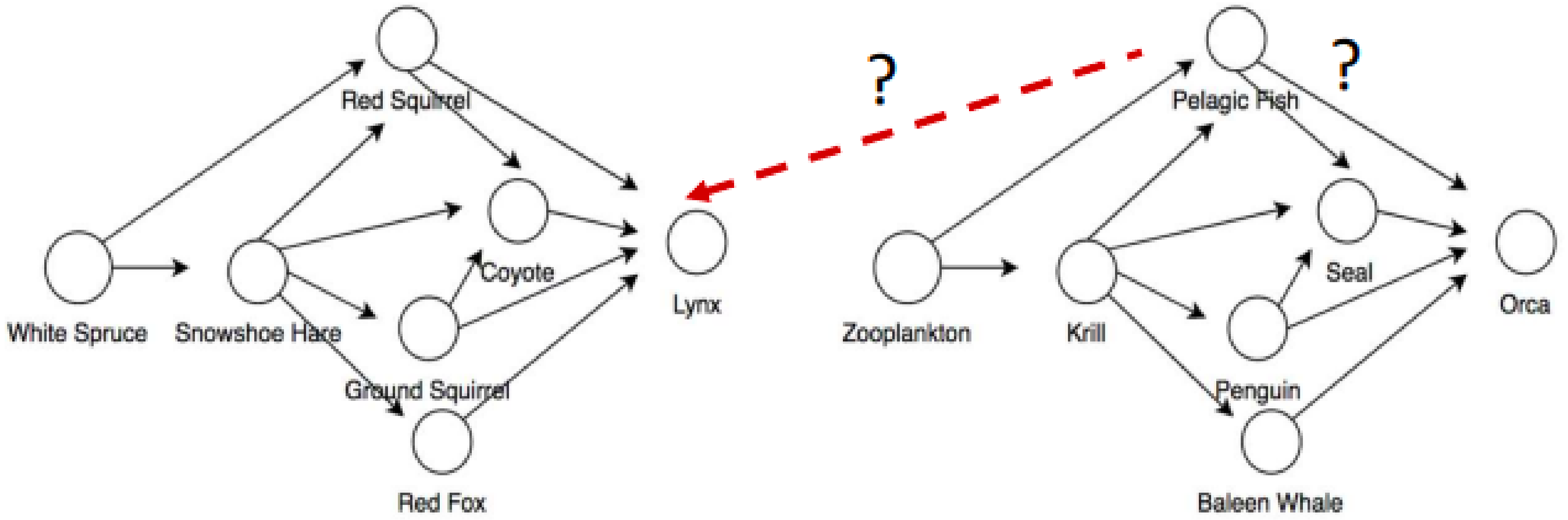


Figure from section 5.4 of the reading

• Common subtrees can cause problems:



Query: Which one is more likely the predator of Pelagic Fish, Lynx or Orca?

Subtree for Lynx and Orca is the same, their representations are the same

Figure from section 5.4 of the reading

Attributed Regular Graphs and WL test

- Given some graph $G = (V, \epsilon, X)$
- Define an equivalence relation: $v \sim u \Leftrightarrow X_v = X_u$
- This allows us to partition the nodes as follows:

$$V = \bigcup_{i \in I} V_{v_i} \qquad V_{v_i} = \{u \in V : X_u = X_{v_i}\}$$

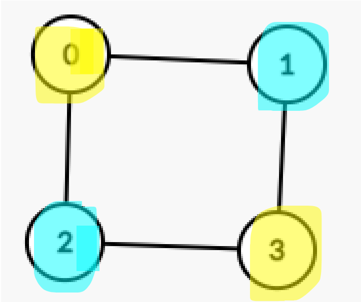
- I Is the indexing set of the classes in the partition

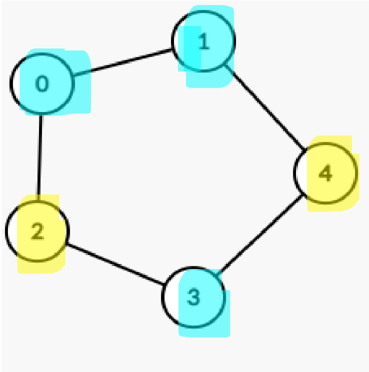
- A graph is an 'Attributed Regular Graph' if:
- For any two classes in the partition, V_{v_i}, V_{v_j} and any two elements $x, y \in V_{v_i}$

$$|\{u \in V_{v_j} : (u,x) \in \epsilon\}| = |\{u \in V_{v_j} : (u,y) \in \epsilon\}|$$

\# of neighbours of x, in class j

\# of neighbours of y, in class j

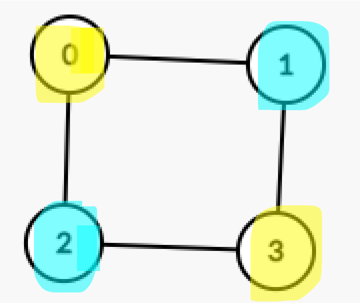




- Define the 'Attributed Graph Representation' to be :
- the partitior $V = \bigcup_i V_{v_i}$, the corresponding attributes $\{X_{v_i} : i \in I\}$ and # of co $\{C_{i,j} : i, j \in I\}$ veen classes

- Where Ci,j is # connections between elements in V_{v_i}, V_{v_j}

- This is a summary, representation does not describe the entire graph



$V = V_{v_0} \bigcup V_{v_1}$ $X_{v_0} =$ $X_{v_1} =$

$C_{0,1} = C_{1,0} = 2$

- Theorem: WL-Test cannot distinguish between graphs with same Attributed Graph Representation

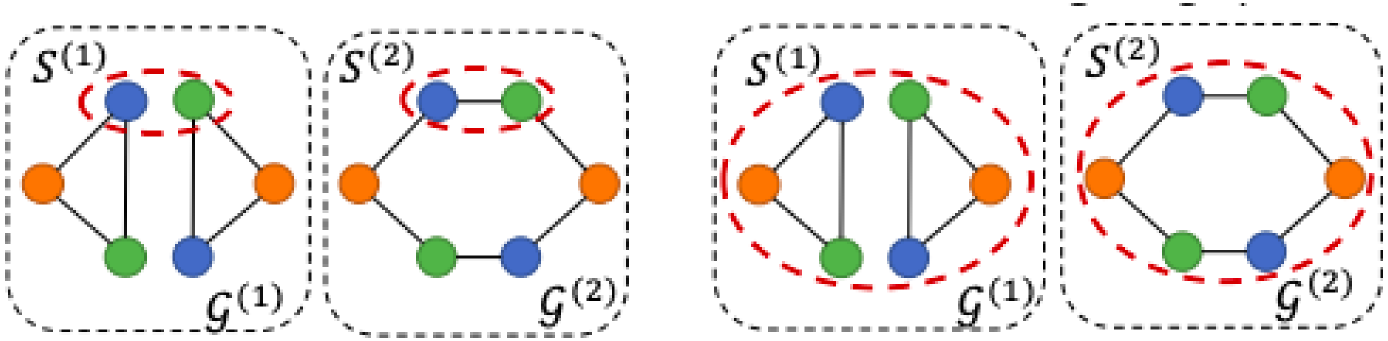


Fig. 5.11: A pair of attributed regular graphs $\mathcal{G}^{(1)}, \mathcal{G}^{(2)}$ with the same configuration and a proper selection of $S^{(1)}$, $S^{(2)}$: MP-GNN and the 1-WL test fail to distinguish $(\mathcal{G}^{(1)}, S^{(1)})$, $(\mathcal{G}^{(2)}, S^{(2)})$.

Figure from section 5.4 of the reading

Injecting random attributes

- Issues arise from not keeping track of indices
- Solution: Modify features to add index to each node

$$augment(V, \epsilon, X) = (V, \epsilon, (X, I))$$

(Add identity matrix as indexing feature)

- Info is then available to detect when cycles begin etc
- Loses permutation invariance
  - Augment operation maps isomorphic graphs to non-isomorphic graphs

Losing Permutation Invariance

For some graph G with adjacency matrix, feature matrix and model weights: A, X, W

Consider a permutation equivariant layer: $P\sigma(AXW) = \sigma(\underbrace{PAP^T}_{A'}\underbrace{PX}_{X'}W)$

Where P is a permutation matrix and $\oplus$ denotes concatenation

Operating on two augmented isomorphic graphs:

$$\sigma(PAP^T P(X \oplus I)W) \neq \sigma(PAP^T((PX) \oplus I)W)$$

The following are isomorphic: $A', X' \cong A; X$

How to get permutation invariance back?

- With some single set of random attributes, a model trained will not be permutation invariant
- Continually train with multiple random augmented attributes
- In expectation we get permutation invariance
- Randomly augmented isomorphic graphs have the same expectation, but each individual sample allows for differentiation between nodes

Relational Pooling (RP-GNN)

- Random attributes are some permuted identity matrix (one hot)
  - Uniformly sample random permutations I_π

$$RP.GNN(V, \epsilon, X := E(MP.GNN(V, \epsilon, (X, I_\pi)))$$

- They show RP-GNN is strictly more powerful than MP-GNN

- RP-GNN is theoretical, we need to approximate with sample
  - Could suffer from high variance / approximation

Murphy, R., Srinivasan, B., Rao, V., & Ribeiro, B. (2019, May). Relational pooling for graph representations. In *International Conference on Machine Learning* (pp. 4663-4673). PMLR.

Random Graph Isomorphic Network (rGIN),

(Sato et al, 2021)

- Choose random features for each node independently from a discrete uniform distribution with p possible values values
    - Multiple nodes can share same feature
    - More flexible – augmentation does not depend on graph size
    - Denote graph augmentation function according to the above by g_{Z_r}

$$rGIN := E(MP.GNN \circ g_{Z_r})$$

Sato, R., Yamada, M., & Kashima, H. (2021). Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM international conference on data mining (SDM)* (pp. 333-341). Society for Industrial and Applied Mathematics.

Abboud, R., Ceylan, I. I., Grohe, M., and Lukasiewicz, T. The surprising power of graph neural networks with random node initialization. arXiv preprint arXiv:2010.01179, 2020.

Theorem 5.6. *(Theorem 4.1 (Abboud et al, 2020)) Consider any invariant mapping $f^* : \mathcal{G}_n \rightarrow \mathbb{R}$, where $\mathcal{G}_n$ contains all graphs with n nodes. Then, there exists a rGIN $f_{MP-GNN} \circ g_{Z_r}$ such that*

$$p(|f_{MP-GNN} \circ g_{Z_r} - f^*| < \varepsilon) > 1 - \delta, \text{ for some given } \varepsilon > 0,\ \delta \in (0,1).$$

(For any choice of epsilon and delta)

Probalistic version of universal approximation!

Conclusion / Questions

- Additional works consider choosing features specific to the graph
  - (see the rest of section 5.4)