

# Topological and Geometric Deep Learning

Research Project

**Student:** Nikolay Chechulin, DSBA-201

**Academic Supervisor:** Oleg Kachan, Research assistant at  
International Laboratory for Applied Topology and Applications

Data Science and Business Analytics  
Faculty of Computer Science



NATIONAL RESEARCH  
UNIVERSITY

June 4, 2022

# Table of Contents

Subject Area

# Overview

## Subject Area

Clique

Incidence Matrix

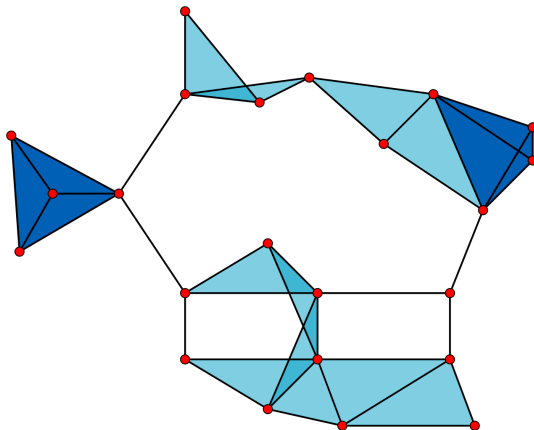
Laplacian matrix

Convolutional Graph Network

Graph Attention Network

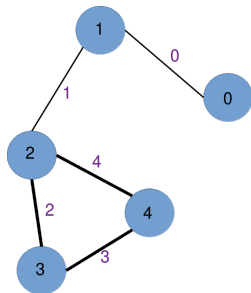
# Clique

A graph clique is a subset of its nodes such that it is fully connected. We will work mainly with 3-cliques, and often will refer to them as triangles.



# Incidence Matrix

If we have an undirected graph  $(V, E)$ , its incidence matrix  $\nabla$  of size  $|V| \times |E|$  such that  $A_{i,j} = 1$  if  $i$ -th vertex is a vertex of  $j$ -th edge. It shows relations between nodes and edges.



$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

# Laplacian matrix

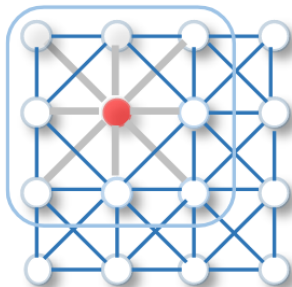
Another matrix representation of a graph. Usually is calculated using the following formula:

$$L_{i,j} = \begin{cases} \deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise,} \end{cases}$$

However, other definitions also take place:  $L = D - A$ , where  $D$  is a degree matrix and  $A$  is an adjacency matrix. Another way to calculate a Laplacian is  $L = \nabla \nabla^T$ , where  $\nabla$  is an incidence matrix.

# Convolutional Graph Network I

A type of GNN which generalizes the convolution operation to graphs.



Convolution on image



Convolution on graph

# Convolutional Graph Network II

Assume we have a graph of  $N$  nodes, where each node has  $F$  features. We can construct an  $N \times F$  matrix called feature matrix. The first layer takes the feature matrix, and performs the following operation:

$Z = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X W$ , where:

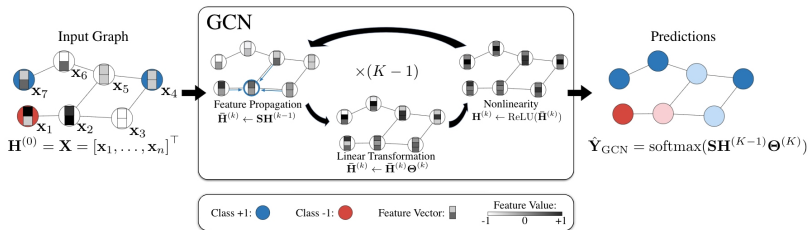
- ▶  $Z$  is resulting  $N \times C$  signal
- ▶  $D$  is  $N \times N$  degree matrix
- ▶  $A$  is  $N \times N$  adjacency matrix with self-loops
- ▶  $X$  is  $N \times F$  feature matrix (input signal)
- ▶  $W$  is  $F \times C$  learnable weight matrix

The last (output) layer usually applies *softmax* function to each row resulting in a new matrix  $S$ . Then, in order to classify a node  $v_i$  we simply take the index of maximum of  $S_i$ .



# Convolutional Graph Network III

The architecture of a graph convolutional network is presented on the figure below.



**Figure:** Architecture of a GCN

# Graph Attention Network I

A type of GNN which uses attention mechanism (also borrowed from ‘casual’ neural networks) which allows us to work with inputs of variable sizes and to focus on the most important features. The attention mechanism is a function  $a : \mathbb{R}^C \times \mathbb{R}^C \rightarrow \mathbb{R}$  which takes two feature vectors  $X_i, X_j$  and returns a scalar representing how tight the connection between  $v_i$  and  $v_j$  is.

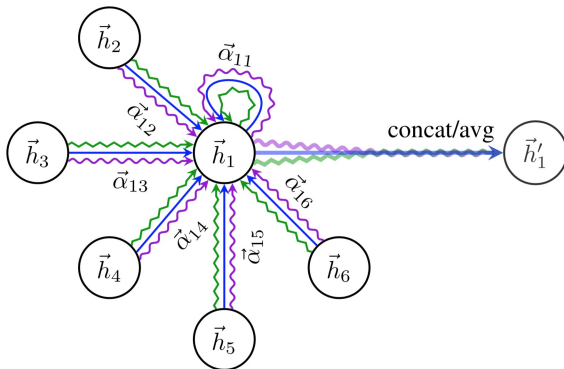
# Graph Attention Network II

We introduce an  $N \times N$  matrix  $\mathbf{e}$  storing the attention between the nodes:  $e_{i,j} = a(W \cdot X_i, W \cdot X_j)$ .

**Don't calculate all pairwise attentions!** One suggested solution is to use a neighborhood  $\mathcal{N}_i$  of a vertex  $v_i$  and then compute the attentions between  $v_i$  and its' neighbors. Existing models uses neighborhood of size 1, and perform great.

One might also want to normalize the coefficients. In order to do that, we can apply softmax function:  $c_{i,j} = \text{softmax}_j(e_{i,j})$ .

# Graph Attention Network III



**Figure:** An example of multi-head attention in a neighborhood of size 1

# References I

- [1] Vijay Prakash Dwivedi et al. “Benchmarking Graph Neural Networks”. In: *CoRR* abs/2003.00982 (2020). arXiv: 2003.00982. URL: <https://arxiv.org/abs/2003.00982>.
- [2] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *arXiv:1609.02907 [cs, stat]* (Feb. 2017). arXiv: 1609.02907. URL: <http://arxiv.org/abs/1609.02907>.
- [3] Yao Ma and Jiliang Tang. *Deep Learning on Graphs*. Cambridge University Press, 2021. DOI: 10.1017/9781108924184.
- [4] Petar Veličković et al. “Graph Attention Networks”. In: *arXiv:1710.10903 [cs, stat]* (Feb. 2018). arXiv: 1710.10903. URL: <http://arxiv.org/abs/1710.10903>.
- [5] Felix Wu et al. “Simplifying Graph Convolutional Networks”. In: *arXiv:1902.07153 [cs, stat]* (June 2019). arXiv: 1902.07153. URL: <http://arxiv.org/abs/1902.07153>.

## References II

- [6] Zonghan Wu et al. “A Comprehensive Survey on Graph Neural Networks”. In: *IEEE transactions on neural networks and learning systems* 32.1 (Jan. 2021), pp. 4–24. ISSN: 2162-2388. DOI: [10.1109/TNNLS.2020.2978386](https://doi.org/10.1109/TNNLS.2020.2978386).
- [7] Keyulu Xu et al. “How Powerful are Graph Neural Networks?”. In: *CoRR* abs/1810.00826 (2018). arXiv: [1810.00826](https://arxiv.org/abs/1810.00826). URL: <http://arxiv.org/abs/1810.00826>.

Any questions?