

Lecture 2: Graph neural networks

Katsiaryna Haitsiukevich, Alison Pouplin, Çağlar Hızlı, Erik Schultheis,
Vikas Garg

Housekeeping

General

- Course enrollment / confirming participation, [instructions](#)
- GNN exercise is released today.
- **DL for the GNN exercise** (Wed 6.11, 23:00)
- Solution session for the exercise (Thu 7.11)
 - Be prepared to present your solutions!
- Preliminary: the paper list is released on Friday (1.11)
- Preliminary: **DL for selecting paper preferences on Tue (11.11)**

In the previous episode of GDL...

Recap of the previous lecture

- Statistical learning theory
 - Assumptions for learning in high-dimensional spaces

Recap of the previous lecture

- Statistical learning theory
 - Assumptions for learning in high-dimensional spaces
- Perspective on model inputs as signals on geometric domain
 - Exploiting the underlying structure of the data to set assumptions

Recap of the previous lecture

- Statistical learning theory
 - Assumptions for learning in high-dimensional spaces
- Perspective on model inputs as signals on geometric domain
 - Exploiting the underlying structure of the data to set assumptions
- Group theory
 - Mathematical tool to work with symmetries of the data structure

Today's lecture

- Sets and Graphs

Today's lecture

- Sets and Graphs
- Invariance and Equivariance

Today's lecture

- Sets and Graphs
- Invariance and Equivariance
- Graph Neural Networks (GNNs)

Today's lecture

- Sets and Graphs
- Invariance and Equivariance
- Graph Neural Networks (GNNs)
- Geometric Deep Learning Blueprint

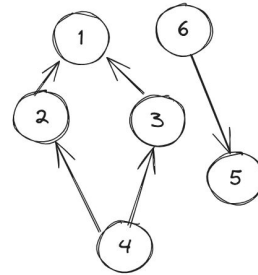
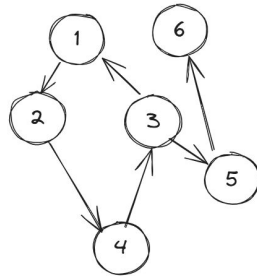
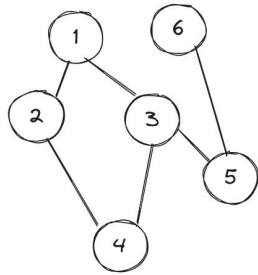
Graphs: essentials

What is a graph?

A graph is an unordered tuple of two sets

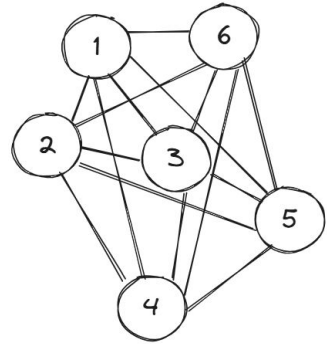
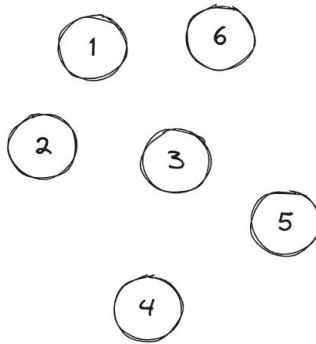
$$\mathcal{G} = (V, E)$$

- Set of nodes (vertices) V
- Set of edges 2-tuples (encoding relations) $E \subseteq (V \times V)$
- And optionally global attributes of a graph can be present



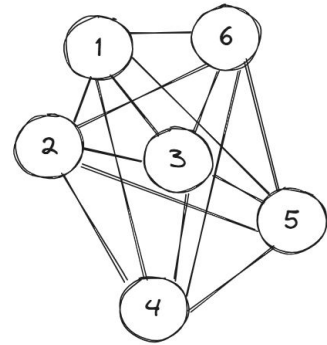
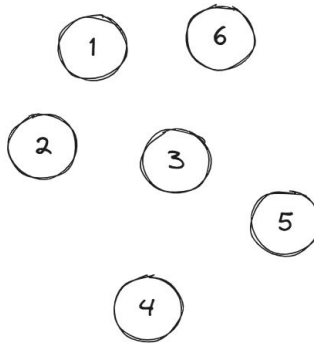
A special case of a graph: a set

- Graph without edges
- Real world example: points clouds for object representations
- However, not many of the problems can be characterised by sets only
- We need to model connections!



A special case of a graph: a fully-connected graph

- Graph where every pair of vertices is connected with an edge
- A synonym: a complete graph
- Real world example: small groups of friends

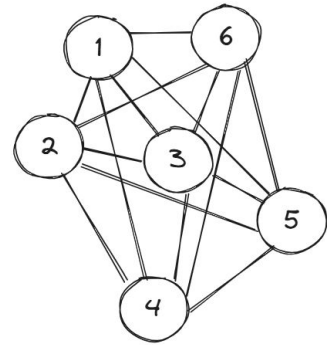
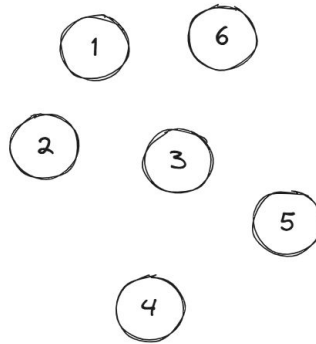


A special case of a graph: a fully-connected graph

- Graph where every pair of vertices is connected with an edge
- A synonym: a complete graph
- Real world example: small groups of friends

Question:

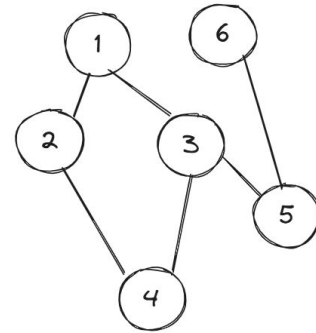
If the connectivity of a graph is unknown, is it the best to model it as a set or as a fully-connected graph?



Adjacency matrix: graph connectivity

- The connectivity information of a graph can be represented as a quadratic matrix called adjacency matrix
- Adjacency matrix consists of 0 and 1
- Value 1 in row k and column m corresponds to an between vertices k and m

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

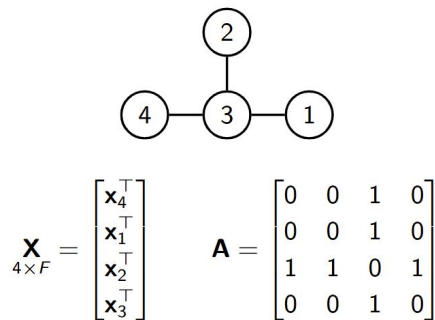
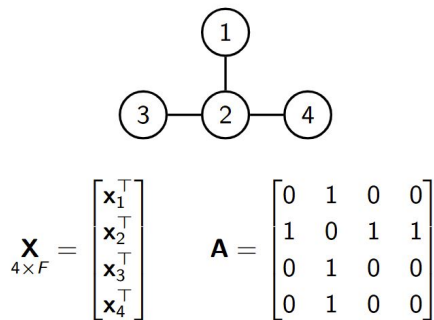


Graph representations

- *How can a graph be represented?*

Graph representations

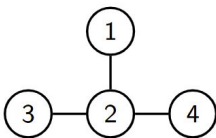
- Features of the nodes
- Connectivity information
- Optionally: edges features and global features

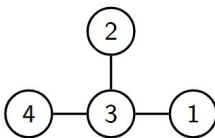


Graph representations

- Features of the nodes
- Connectivity information
- Optionally: edges features and global features

*Two different representations
or the same object make
modelling tricky*


$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \\ \mathbf{x}_4^T \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

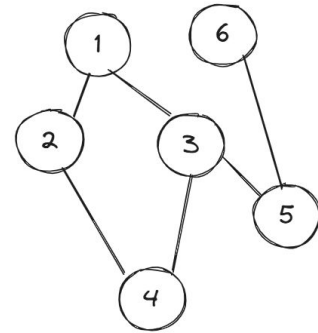

$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_4^T \\ \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Neighborhoods and degree of a node

- A neighborhood of a node

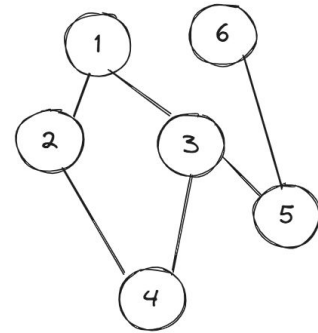
is a set of vertices the node is connected to by an edge

What is the neighborhood of the node number 3?



Neighborhoods and degree of a node

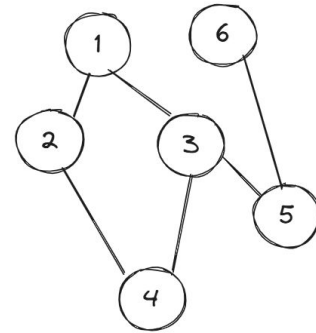
- A neighborhood of a node
is a set of vertices the node is connected to by an edge
- A degree of a node
is the number of its neighbors



Neighborhoods and degree of a node

- A neighborhood of a node
is a set of vertices the node is connected to by an edge
- A degree of a node
is the number of its neighbors

*What is the degree of the
node number 3?*



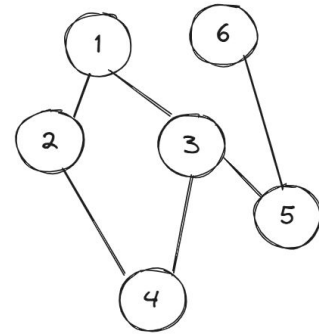
Laplacian matrix

Laplacian matrix is defined as a difference between

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

- a diagonal degree matrix (*each element is a degree of the corresponding node*) and
- an adjacency matrix

$$\mathbf{D} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$



Importance of Laplacian matrix

- Laplacian matrix encodes the same information as Adjacency matrix
- Both are used

Importance of Laplacian matrix

- Laplacian matrix encodes the same information as Adjacency matrix
- Both are used
- Laplacian matrix exhibits desired properties in eigendecomposition
 - The matrix of size $n \times n$ has n independent eigenvectors
 - All eigenvalues are real and non-negative
 - The smallest eigenvalue is always zero
 - The eigenvalues indicate connected components in the graph (clusters)

Importance of Laplacian matrix

- Laplacian matrix encodes the same information as Adjacency matrix
- Both are used
- Laplacian matrix exhibits desired properties in eigendecomposition
 - The matrix of size $n \times n$ has n independent eigenvectors
 - All eigenvalues are real and non-negative
 - The smallest eigenvalue is always zero
 - The eigenvalues indicate connected components in the graph (clusters)
- Fourier transform of a graph
 - A linear combination of graph spectral components
 - Utilizes eigenvectors of the Laplacian (similar to sinusoids of a classical Fourier transform)

Importance of Laplacian matrix

- Laplacian matrix encodes the same information as Adjacency matrix
- Both are used
- Laplacian matrix exhibits desired properties in eigendecomposition
 - The matrix of size $n \times n$ has n independent eigenvectors
 - All eigenvalues are real and non-negative
 - The smallest eigenvalue is always zero
 - The eigenvalues indicate connected components in the graph (clusters)
- Fourier transform of a graph
 - A linear combination of graph spectral components
 - Utilizes eigenvectors of the Laplacian (similar to sinusoids of a classical Fourier transform)
- Operation of convolution is equivalent to multiplication in frequency domain

Motivational examples

Graphs can represent a lot of data

- Molecules
- Social connections: social networks
- Physical systems: modelling interactions
- Roads: journey planning
- Knowledge graphs: wikipedia
- OCR relations extracted from a PDF document
- Solving PDEs

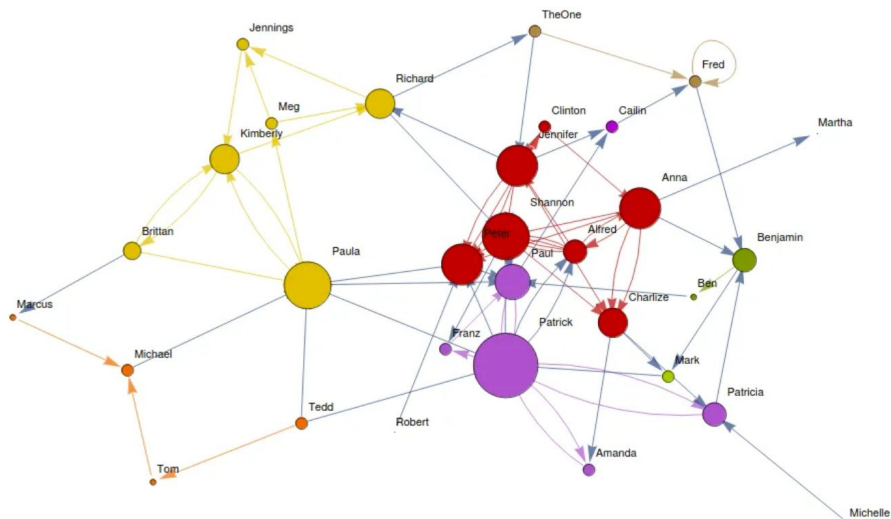


Image source: [Blog post on message-passing](#)

Types of tasks on Graphs

- Node-level tasks
- Edge-level tasks
- Graph-level tasks

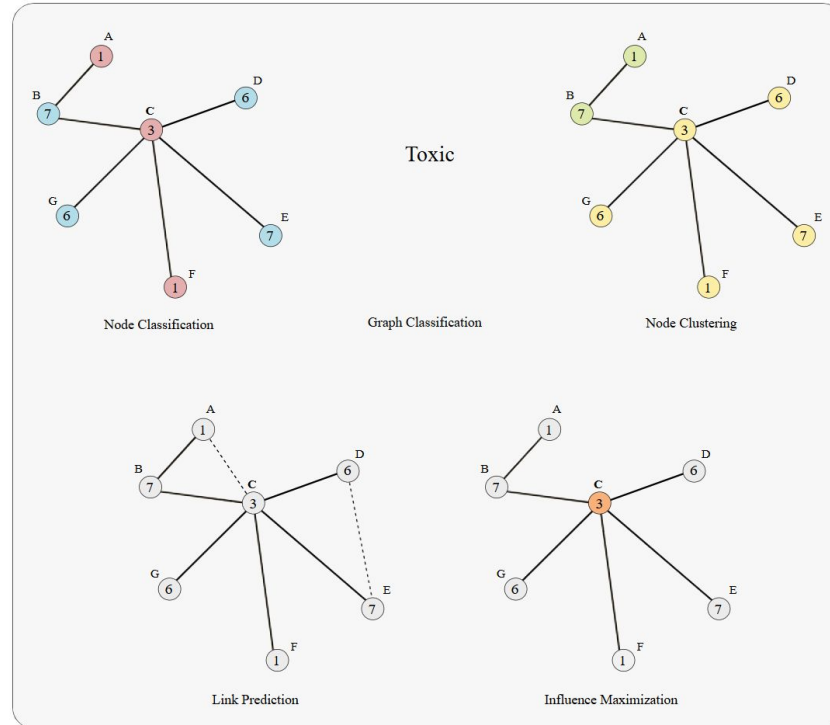


Image source: [Blog post on graph problems](#)

A note on assumptions

- In the general case, the connectivity information of a graph can be unknown
 - It is known but can be suboptimal for the task at hand

A note on assumptions

- In the general case, the connectivity information of a graph can be unknown
 - It is known but can be suboptimal for the task at hand
- Learning the structure of a graph is an exciting area of research
 - beyond the scope of this lecture

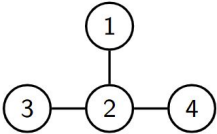
A note on assumptions

- In the general case, the connectivity information of a graph can be unknown
 - It is known but can be suboptimal for the task at hand
- Learning the structure of a graph is an exciting area of research
 - beyond the scope of this lecture

For now: assume that the structure is known and it is sufficient for the tasks!

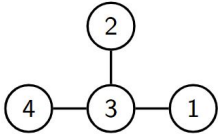
Invariance and Equivariance

Revisiting the problem encountered before: Two graphs can have “different representations”



Graph 1: A central node 2 connected to nodes 1, 3, and 4.

$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \\ \mathbf{x}_4^\top \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

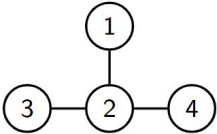


Graph 2: A central node 3 connected to nodes 2, 4, and 1.

$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_4^\top \\ \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

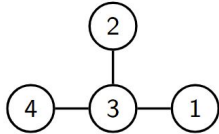
- To learn it directly from data a model needs to see all of the combinations (4! permutations)
-> the data requirements grow significantly

Revisiting the problem encountered before: Two graphs can have “different representations”



Graph 1: A central node 2 connected to nodes 1, 3, and 4.

$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \\ \mathbf{x}_4^\top \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



Graph 2: A central node 3 connected to nodes 2, 4, and 1.

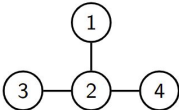
$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_4^\top \\ \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \mathbf{x}_3^\top \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

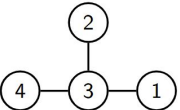
- To learn it directly from data a model needs to see all of the combinations (4! permutations)
-> the data requirements grow significantly

-> we want to exploit the symmetry!

Linking back to statistical learning

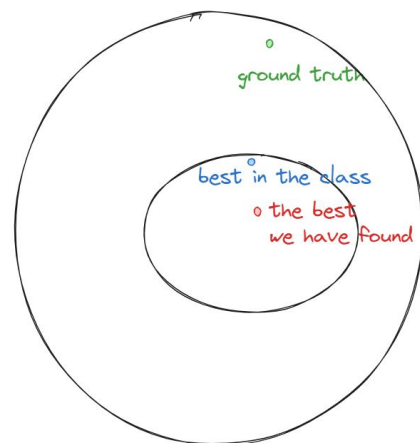
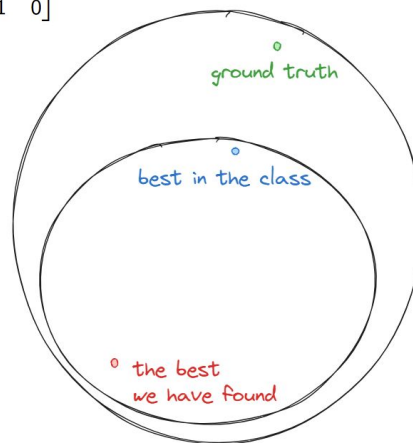
- The assumptions are needed to restrict the class of possible models


$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \\ \mathbf{x}_4^T \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

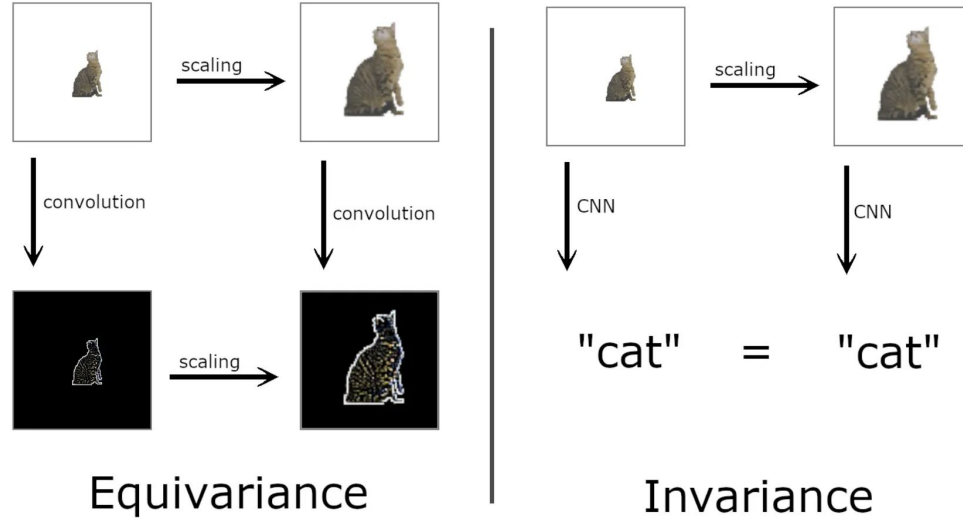

$$\mathbf{X}_{4 \times F} = \begin{bmatrix} \mathbf{x}_4^T \\ \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

We want to exploit symmetry:

With a fixed size of a dataset we can find a better approximation if the search space of possible function is smaller



Invariance and Equivariance



Permutation invariance for sets

There is no natural ordering in the data

-> the output of the model should not depend on the input order.

$$f \left(\begin{array}{c} \text{x}_5 \\ \text{x}_4 \end{array} \begin{array}{c} \text{x}_1 \\ \text{x}_2 \\ \text{x}_3 \end{array} \right) = \mathbf{y} = f \left(\begin{array}{c} \text{x}_2 \\ \text{x}_5 \\ \text{x}_1 \end{array} \begin{array}{c} \text{x}_4 \\ \text{x}_3 \end{array} \right)$$

Symmetry group \mathfrak{S} : n -element Permutation group Σ_n

Group element $g \in \mathfrak{S}$: Permutations

Permutation invariance for graphs

$$f \left(\begin{array}{c} \text{X}_5 \quad \text{X}_1 \\ \text{X}_4 \quad \text{X}_2 \\ \text{X}_3 \end{array} \right) = \mathbf{y} = f \left(\begin{array}{c} \text{X}_2 \\ \text{X}_5 \quad \text{X}_4 \\ \text{X}_1 \quad \text{X}_3 \end{array} \right)$$

$$f \left(\begin{array}{c} \text{X}_5 \text{---} \text{X}_1 \\ \text{X}_5 \text{---} \text{X}_2 \\ \text{X}_4 \text{---} \text{X}_2 \\ \text{X}_4 \text{---} \text{X}_3 \end{array} \right) = \mathbf{y} = f \left(\begin{array}{c} \text{X}_2 \\ \text{X}_5 \text{---} \text{X}_4 \\ \text{X}_1 \text{---} \text{X}_3 \end{array} \right)$$

Permutation equivariance

- *Ok, permutation invariance is needed to account for ordering but why do we need permutation equivariance?*

Permutation equivariance

1. We may actually want the result to change according to the applied transformation, e.g. rotations of a molecule (the coordinates of atoms)

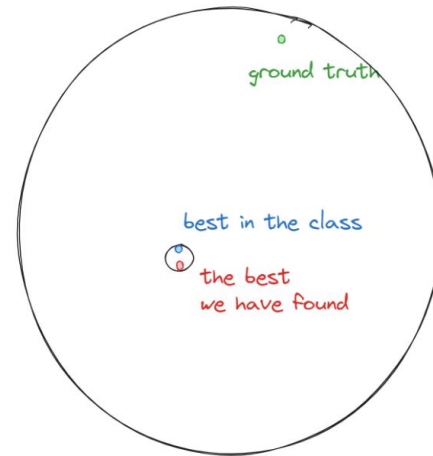
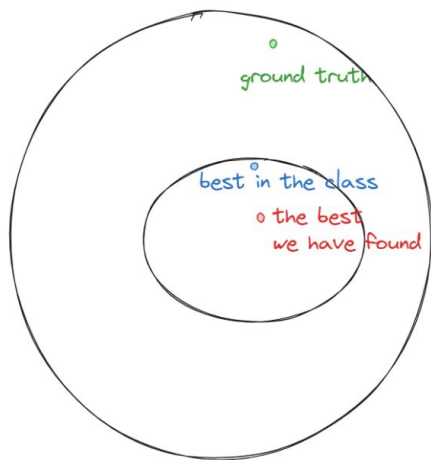
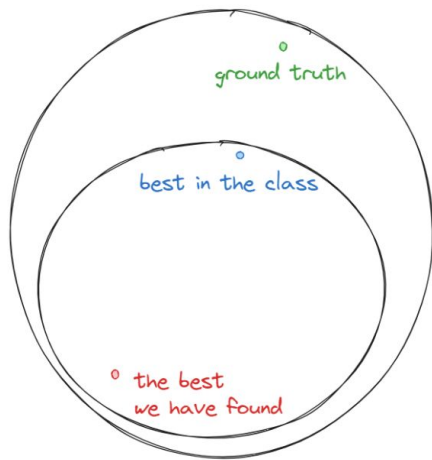
Permutation equivariance

1. We may actually want the result to change correspondingly to the applied transformation
2. We can express a much larger class of transformations by alternating between invariant and equivariant layers

Permutation equivariance

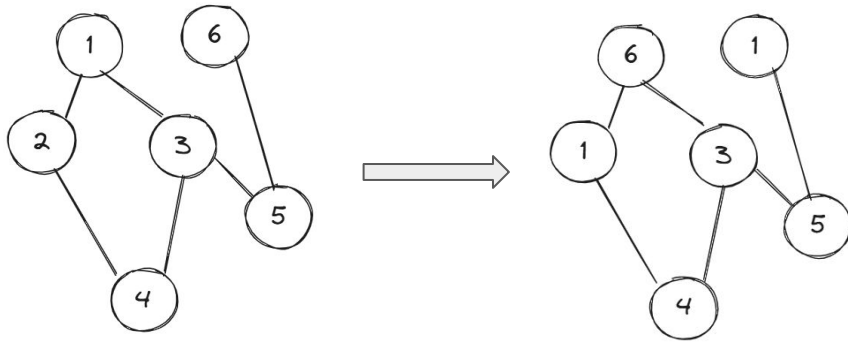
1. We may actually want the result to change correspondingly to the applied transformation
2. We can express a much larger class of transformations by alternating between invariant and equivariant layers
 - a. A class of invariant transformations is quite limited
 - b. Equivariant transformations are more expressive
 - c. By using equivariant + invariant transformation -> the result is invariant, but more expressive
 - d. Can be seen in the GDL blueprint in the end of the lecture (and demonstrated in the exercise)

Linking back to statistical learning



Permutations: Permutation matrix in practice

- Permutation matrix should have 1 in every row and every column



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

Invariance and equivariance

- Sets (P is a permutation matrix)

$$f(\mathbf{PX}) = f(\mathbf{X}) \qquad \mathbf{F}(\mathbf{PX}) = \mathbf{PF}(\mathbf{X})$$

Invariance and equivariance

- Sets (P is a permutation matrix)

$$f(\mathbf{PX}) = f(\mathbf{X}) \qquad \mathbf{F}(\mathbf{PX}) = \mathbf{PF}(\mathbf{X})$$

- Graphs

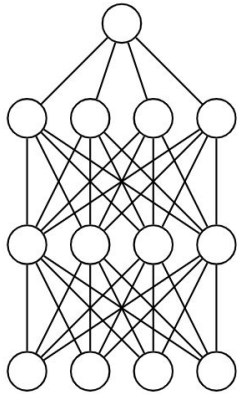
$$f(\mathbf{PX}, \mathbf{PAP}^T) = f(\mathbf{X}, \mathbf{A}) \qquad \mathbf{F}(\mathbf{PX}, \mathbf{PAP}^T) = \mathbf{PF}(\mathbf{X}, \mathbf{A})$$

Are simple MLPs permutation invariant?

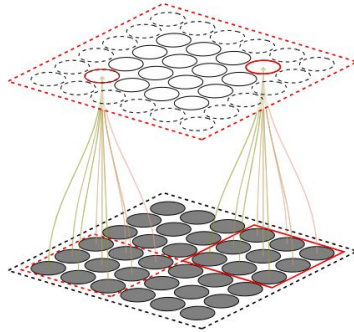
Are simple MLPs permutation invariant?

- no.

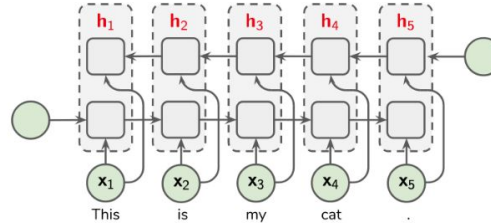
Are any of these architecture permutation invariant / equivariant?



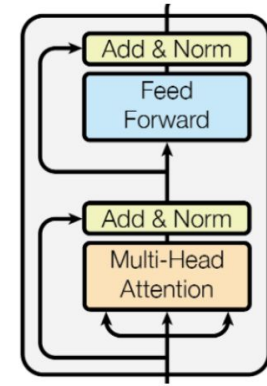
multi-layer
perceptron



convolutional
layer

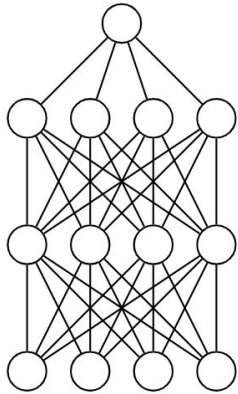


recurrent neural
network

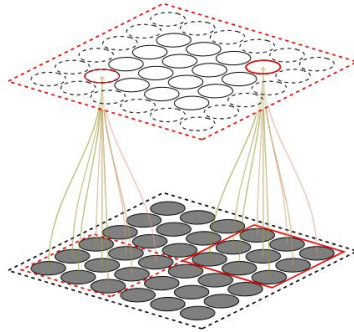


transformer encoder

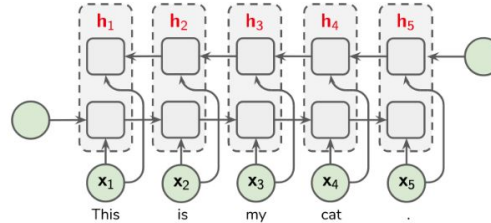
Are any of these architecture permutation invariant / equivariant?



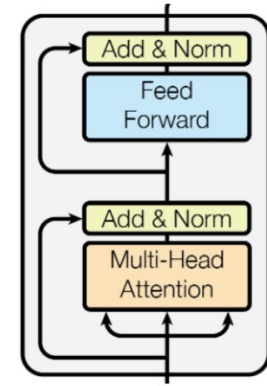
multi-layer
perceptron



convolutional
layer



recurrent neural
network



transformer encoder

Answer: transformer encoder with no positional encoding

Other requirements for GNNs

- Permutation invariance

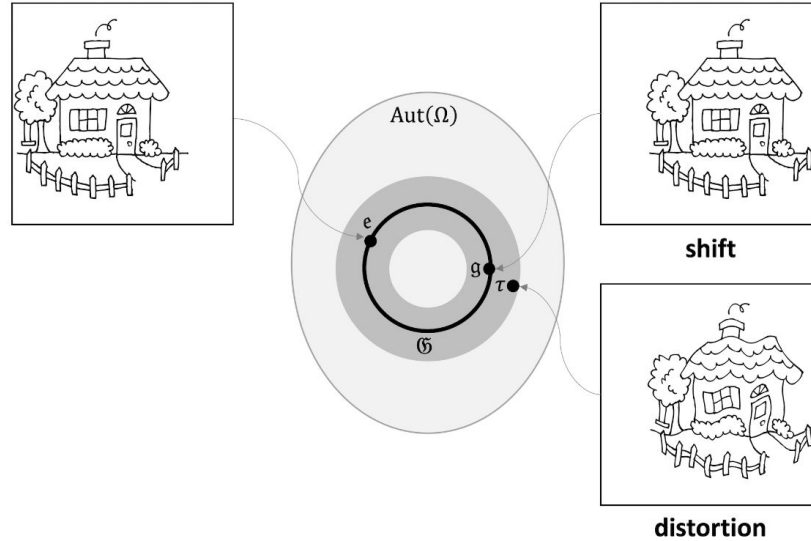
+

- Ability to work with graphs of varying number of nodes
- Ability to take into account the connectivity information

Locality: act on neighborhoods

Caution! Do not overdo it with symmetries! Real world is not perfect.

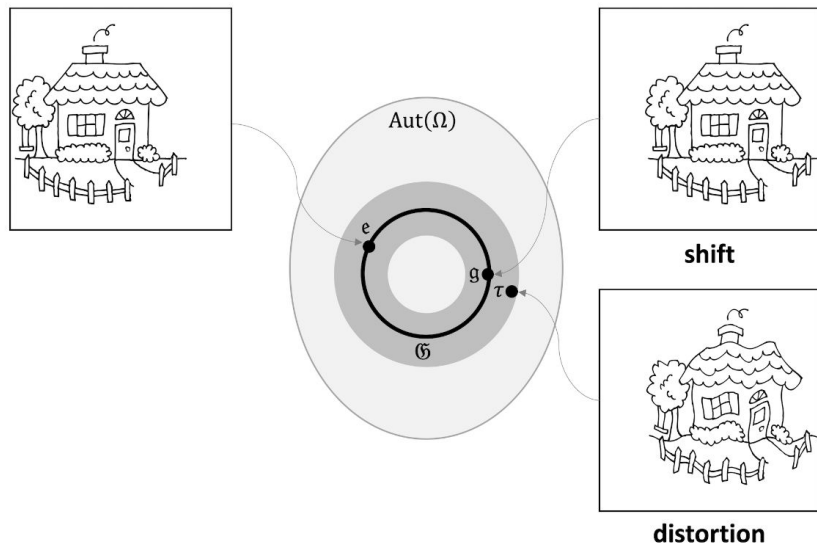
Locality: act on neighborhoods



Locality: act on neighborhoods

Stability under slight deformations

- CNN layers act locally as well
- GNN layers should be local too
- Link back to the statistical learning -> should not make too strong of the assumptions



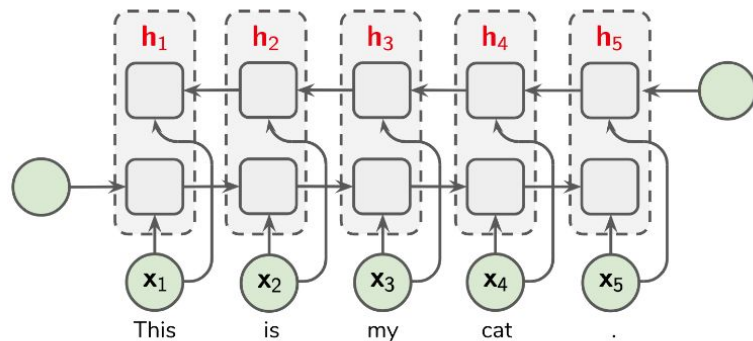
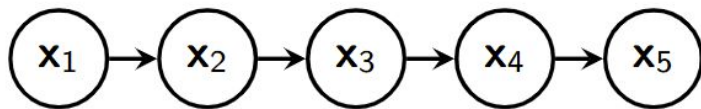
Special graphs

DL architectures: CNN, RNN, Transformer

- Even though there is no permutation equivariance in many existing architectures, these models are still exhibiting some desirable symmetries and can be classified as “special cases of GNNs”.

Processing special graphs: chains

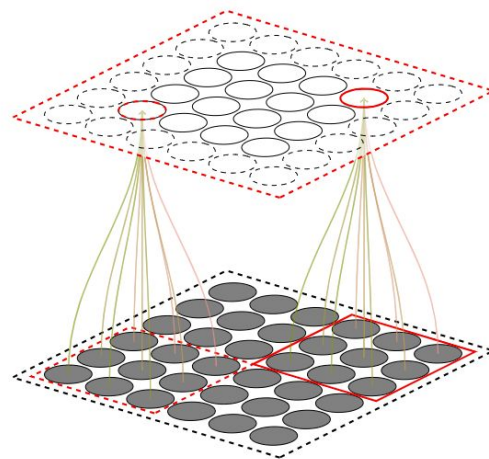
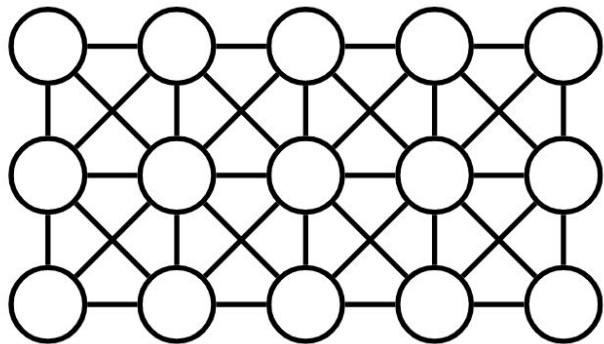
- RNN: a chain graph (physical system modelling)



recurrent neural
network

Processing special graphs: grids

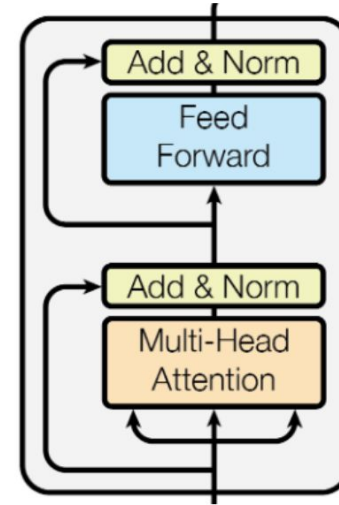
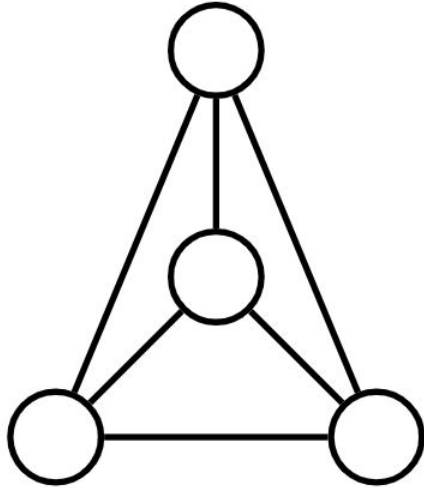
- CNN: a grid graph (PDE solving)



convolutional
layer

Processing special graphs: fully-connected graphs

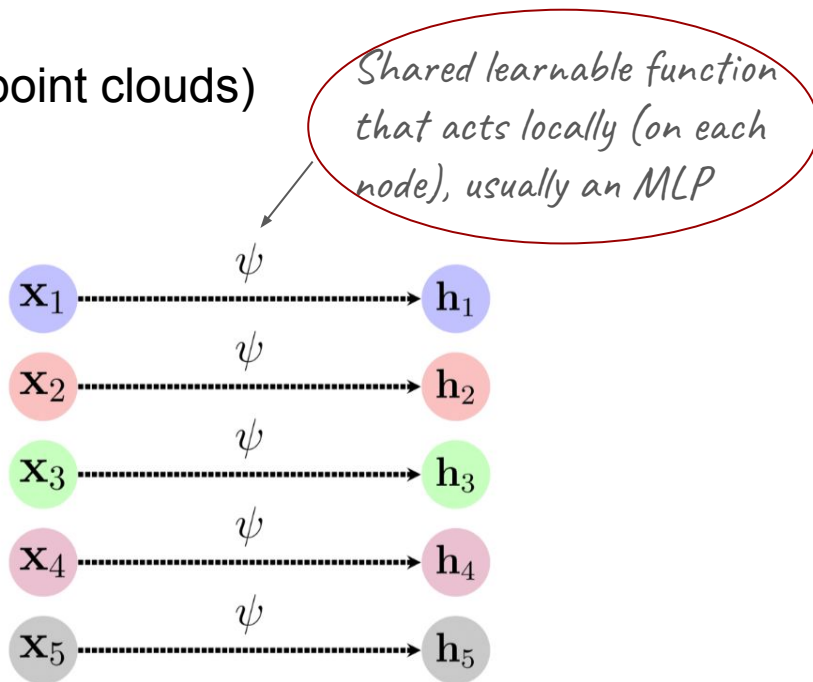
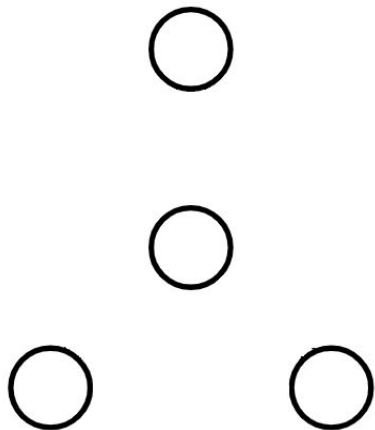
- Transformer encoder: a complete graph (text)



transformer encoder

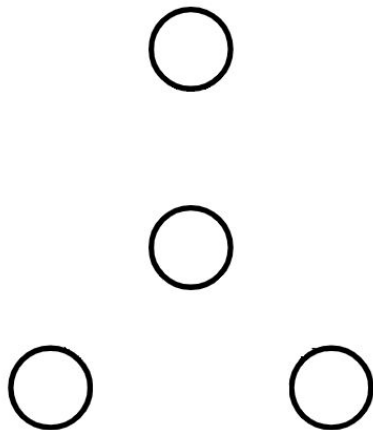
Deep Sets

- Deep sets: a graph with no edges (point clouds)



Deep Sets

- Deep sets: a graph with no edges (point clouds)



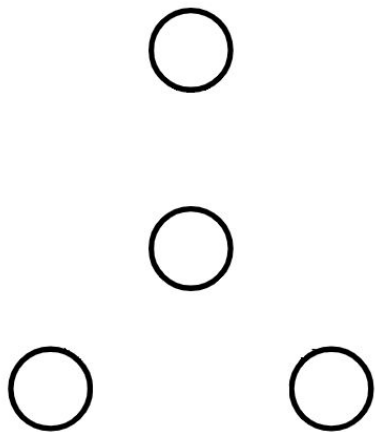
$$f(\mathbf{X}) = \phi \left(\bigoplus_{i \in \mathcal{V}} \psi(\mathbf{x}_i) \right)$$

The diagram illustrates the mathematical formulation of a deep set function. The expression shows a function $f(\mathbf{X})$ that takes a set of points \mathbf{X} as input. It applies a permutation-invariant transformation ϕ to the sum (indicated by \bigoplus) of individual point features $\psi(\mathbf{x}_i)$ for all vertices i in the set \mathcal{V} . Arrows from the text "MLPs" point to the ψ and ϕ functions, indicating they are implemented as multi-layer perceptrons.

*Crucial! Permutation-invariant transformation
for all vertices (sum, avg)*

Deep Sets

- Deep sets: a graph with no edges (point clouds)



Invariant transformation

Equivariant transformation

$$f(\mathbf{X}) = \phi \left(\bigoplus_{i \in \mathcal{V}} \psi(\mathbf{x}_i) \right)$$

*Crucial! Permutation-invariant transformation
for all vertices (sum, avg)*

Beyond special cases: Geometry as a network input

- This is as far as we can get with special cases
- How about processing arbitrary graphs?

Beyond special cases: Geometry as a network input

- This is as far as we can get with special cases
- How about processing arbitrary graphs?

Now the graph structure (geometry) becomes a part of the input!

Beyond special cases: Geometry as a network input

- This is as far as we can get with special cases
- How about processing arbitrary graphs?

Now the graph structure (geometry) becomes a part of the input!

But we still assume that the structure is known.

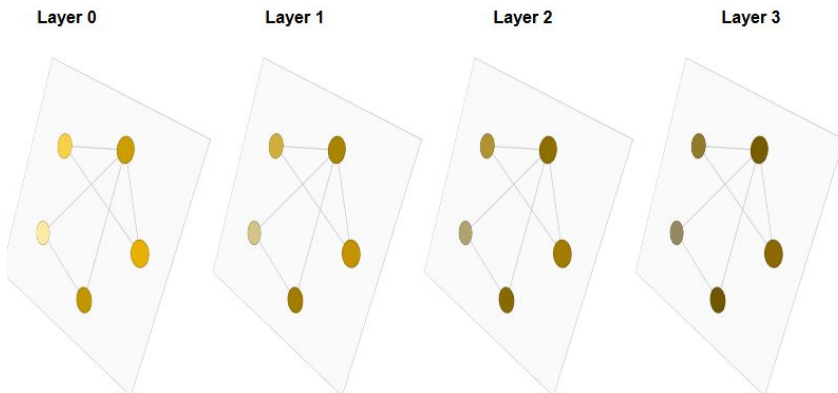
However different general graphs can be processed in the same dataset.

Break -?

Graph Neural Networks

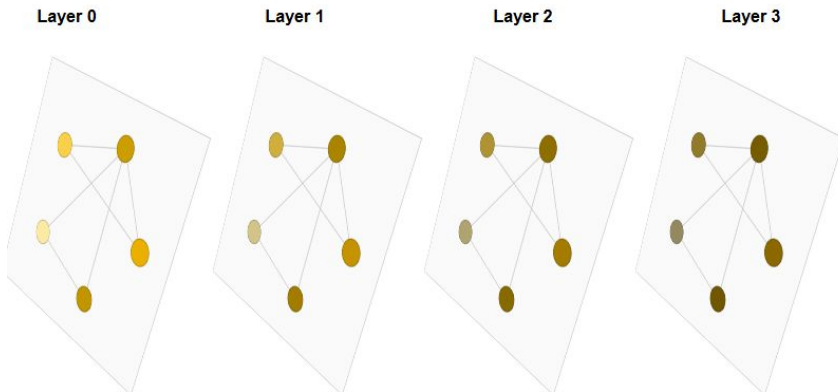
GNNs

- Learnable transformation of graph attributes (node, edge and global features)



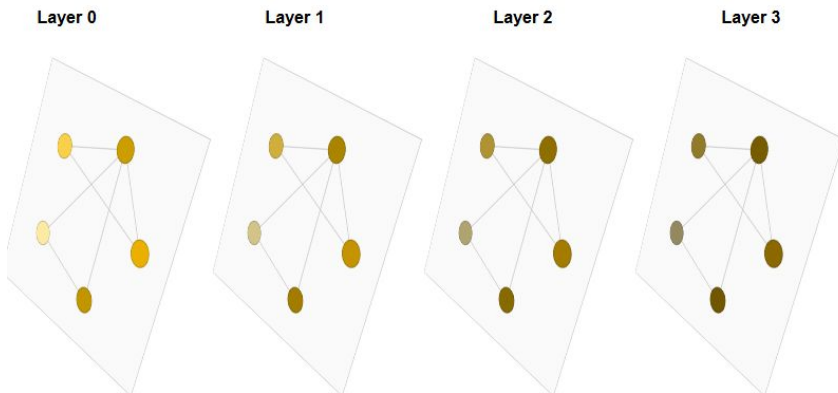
GNNs

- Learnable transformation of graph attributes (node, edge and global features)
- The transformation is permutation invariant



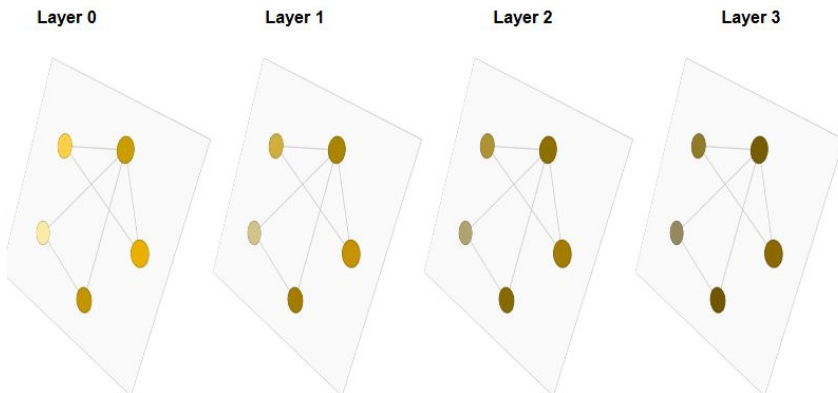
GNNs

- Learnable transformation of graph attributes (node, edge and global features)
- The transformation is permutation invariant
- GNNs operate over graphs (both inputs are outputs are graphs)



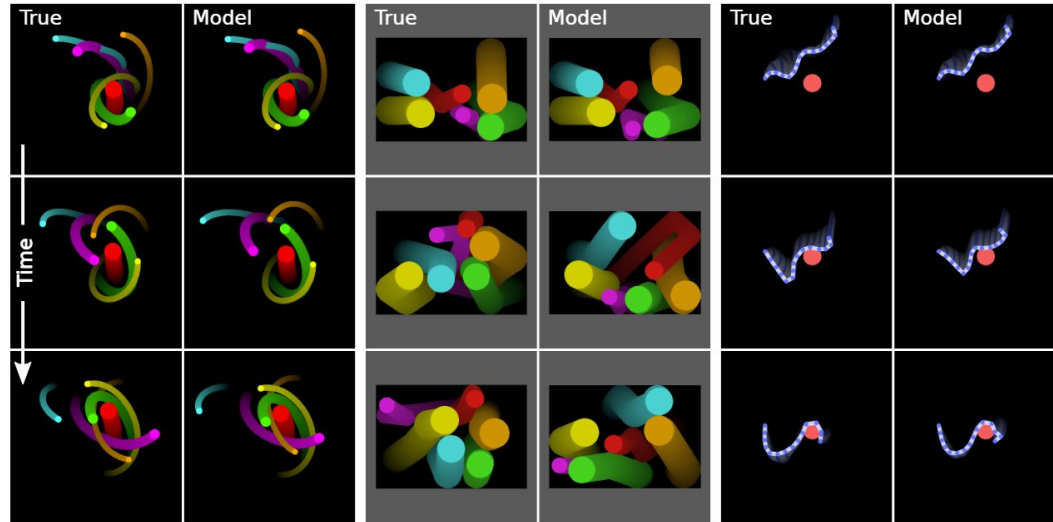
GNNs

- Learnable transformation of graph attributes (node, edge and global features)
- The transformation is permutation invariant
- GNNs operate over graphs (both inputs and outputs are graphs)
- (Usually?) The transformations are applied without changing the structure of the input graph



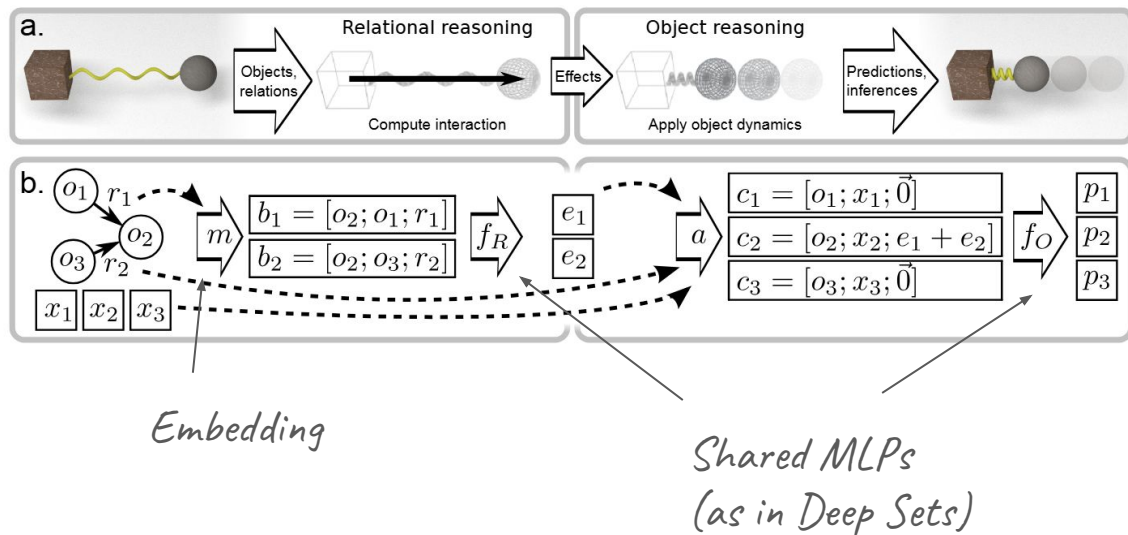
One of the pioneering works: Interaction networks

- Modelling of interactions in physical systems (predict the dynamics)



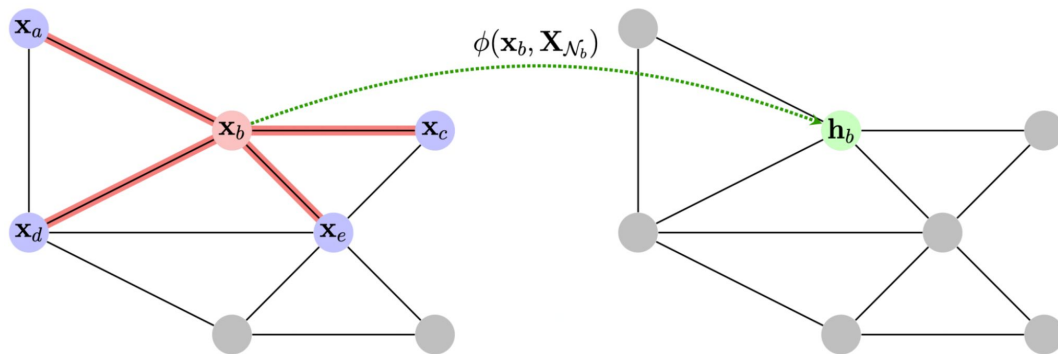
One of the pioneering works: Interaction networks

- Modelling of interactions in physical systems



GNN layers

- Similar idea as in Deep Sets + dependency on the neighborhoods
- The update function is also shared between nodes (locality)



$$X_{N_b} = \{x_a, x_b, x_c, x_d, x_e\}$$

GNN Blueprint

1. Nodes send messages to their neighbors

$$\psi(\mathbf{x}_i, \mathbf{x}_j)$$

2. Messages from the neighbors are aggregated

$$\bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)$$

3. Node features (attributes) are updated based on the received messages


$$\mathbf{h}_i = \phi \left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j) \right)$$

Note! Edge and global features are ignored for simplicity.

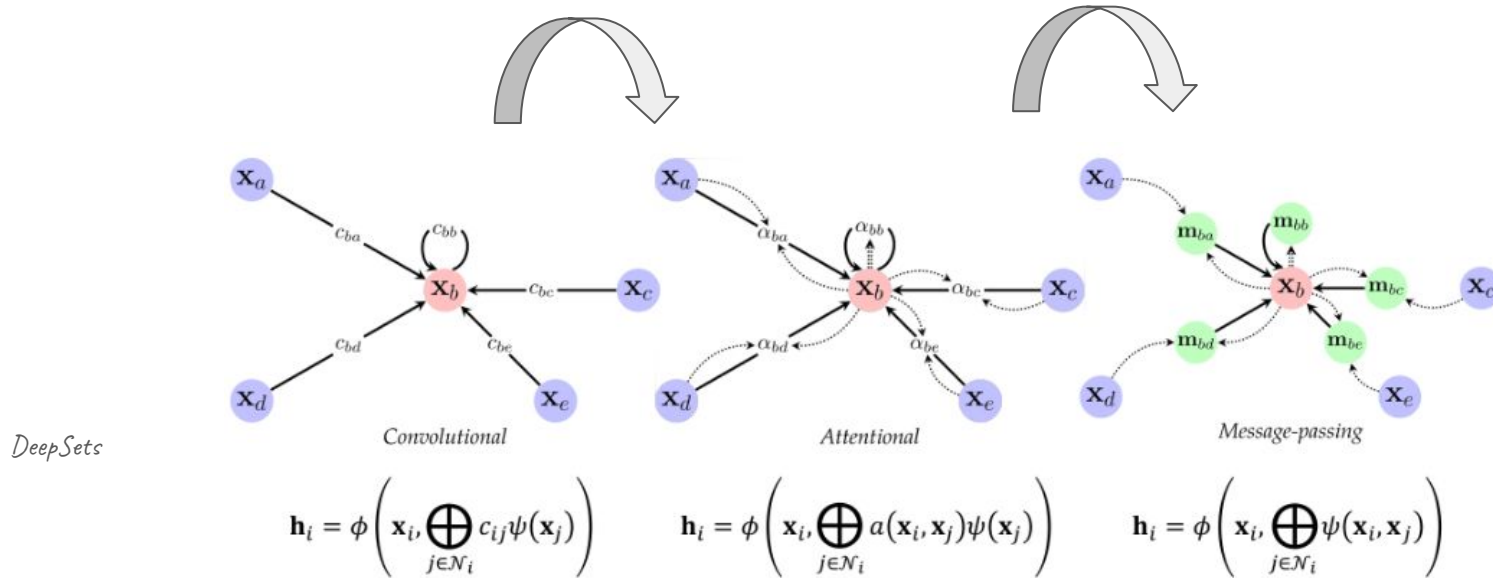
GNN layers

*Permutation
equivariant function*

Local permutation-invariant function


$$\mathbf{F}(\mathbf{X}, \mathbf{A}) = \begin{bmatrix} - & \phi(\mathbf{x}_1, \mathbf{X}_{\mathcal{N}_1}) & - \\ - & \phi(\mathbf{x}_2, \mathbf{X}_{\mathcal{N}_2}) & - \\ & \vdots & \\ - & \phi(\mathbf{x}_n, \mathbf{X}_{\mathcal{N}_n}) & - \end{bmatrix}$$

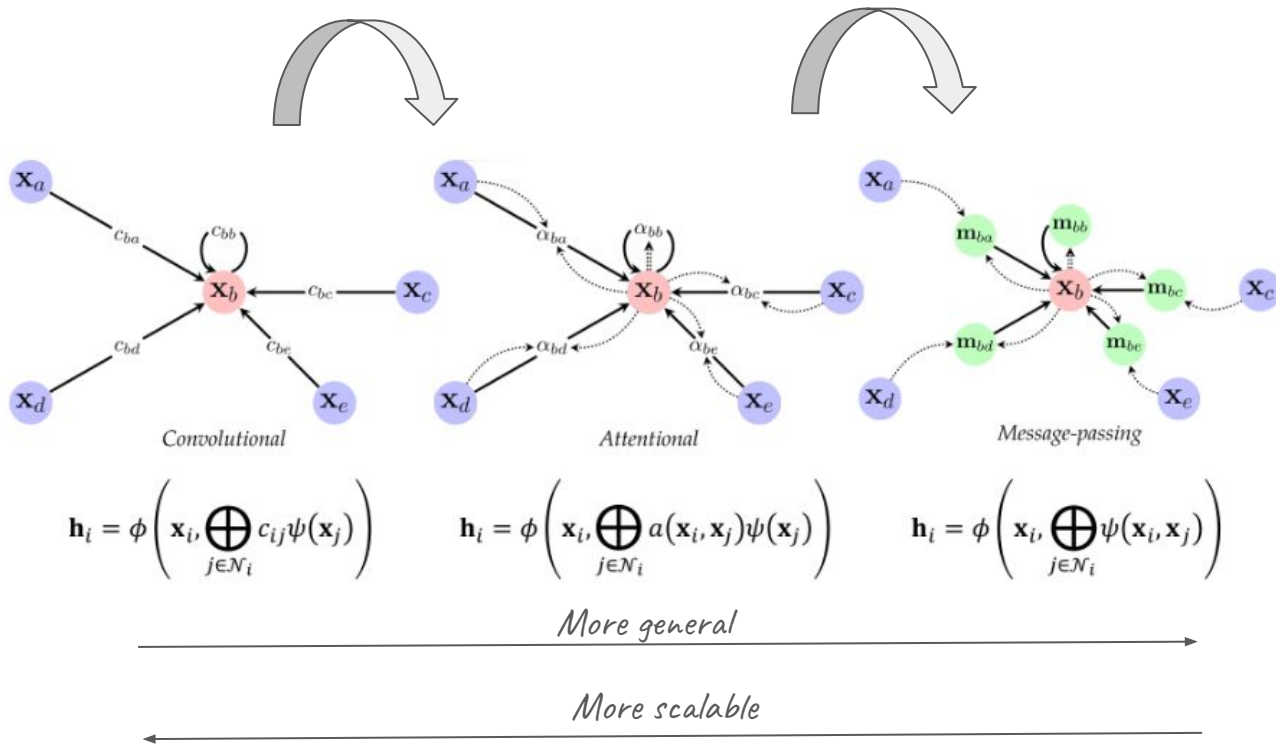
Types of GNNs



The same structure, differences are in how messages are calculated and aggregated

Types of GNNs

Deep Sets



Graph Convolutional network

- Generalizes convolutions for arbitrary structure of graphs beyond grids
- Graph Convolutions are based on spectral perspective of convolutions
- The convolution operation is equivalent to the multiplication in Fourier domain

$$\mathbf{h}_i = \phi \left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij} \psi(\mathbf{x}_j) \right)$$

Graph Convolutional network

- Generalizes convolutions for arbitrary structure of graphs beyond grids
- Graph Convolutions are based on spectral perspective of convolutions
- The convolution operation is equivalent to the multiplication in Fourier domain
 - Scales well with the size of the graph
 - Works well for homophilous graphs
 - Works when the graph is fixed

$$\mathbf{h}_i = \phi \left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij} \psi(\mathbf{x}_j) \right)$$

Graph attentional network (GAT)

- Utilizes attention to calculate weights of neighbors' messages

$$\mathbf{h}_i = \phi \left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j) \psi(\mathbf{x}_j) \right)$$

Graph attentional network (GAT)

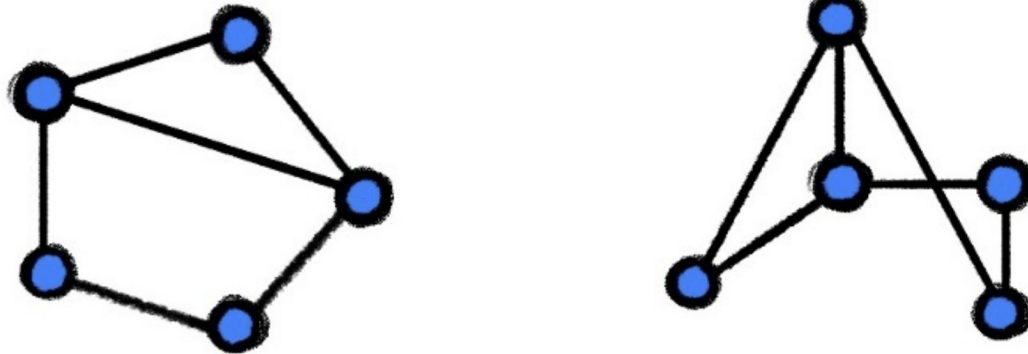
- Utilizes attention to calculate weights of neighbors' messages
 - A bit more general than Graph Convolutional Networks
 - Less scalable than Graph Convolutional Networks
 - Can 'infer' a graph structure, e.g. if the graph is unknown we may pass a fully connected graph. In theory, GAT can learn to put near-zero weights for some of the neighbor messages.

$$\mathbf{h}_i = \phi \left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j) \psi(\mathbf{x}_j) \right)$$

Detour: Expressive power of GNNs

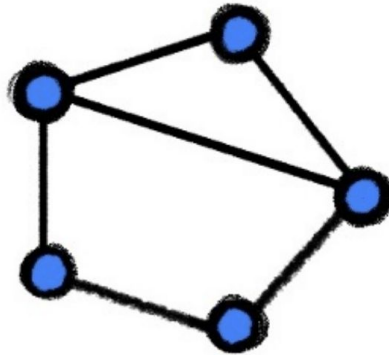
Isomorphic graphs

- Two graphs are isomorphic if there is one-to-one correspondence between their vertices and edges that preserves the connectivity structure of the graphs.

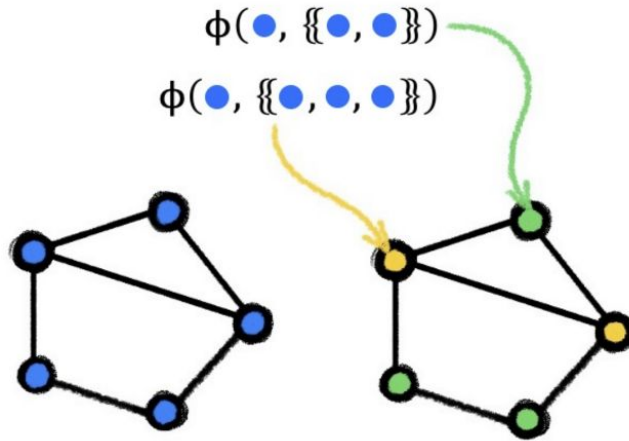


1-WL (Weisfeiler-Lehman) test

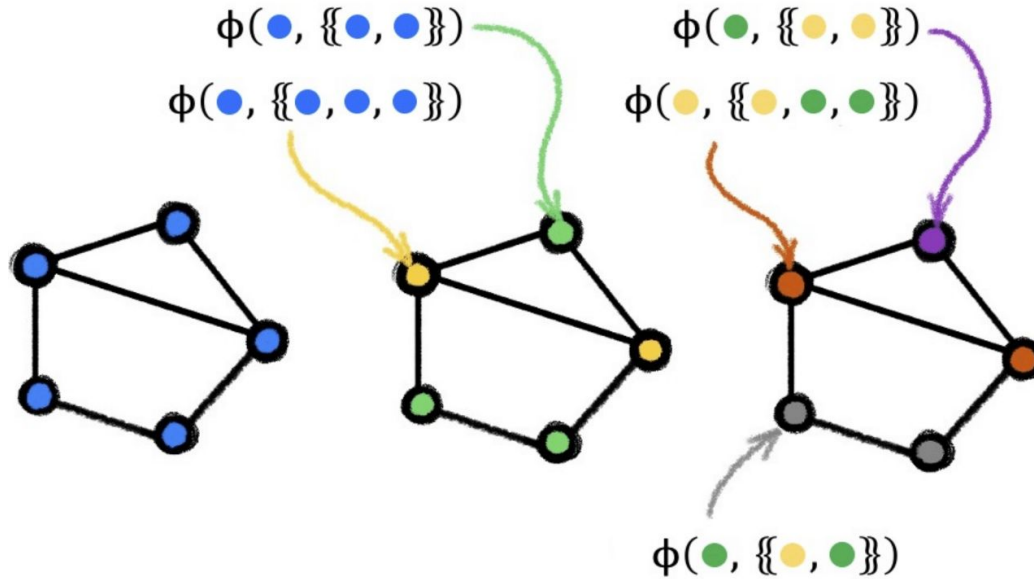
- The test for two graph being **isomorphic**
- Implemented by iteratively tracking neighborhoods that arise and changing their colors (colors = hashes)



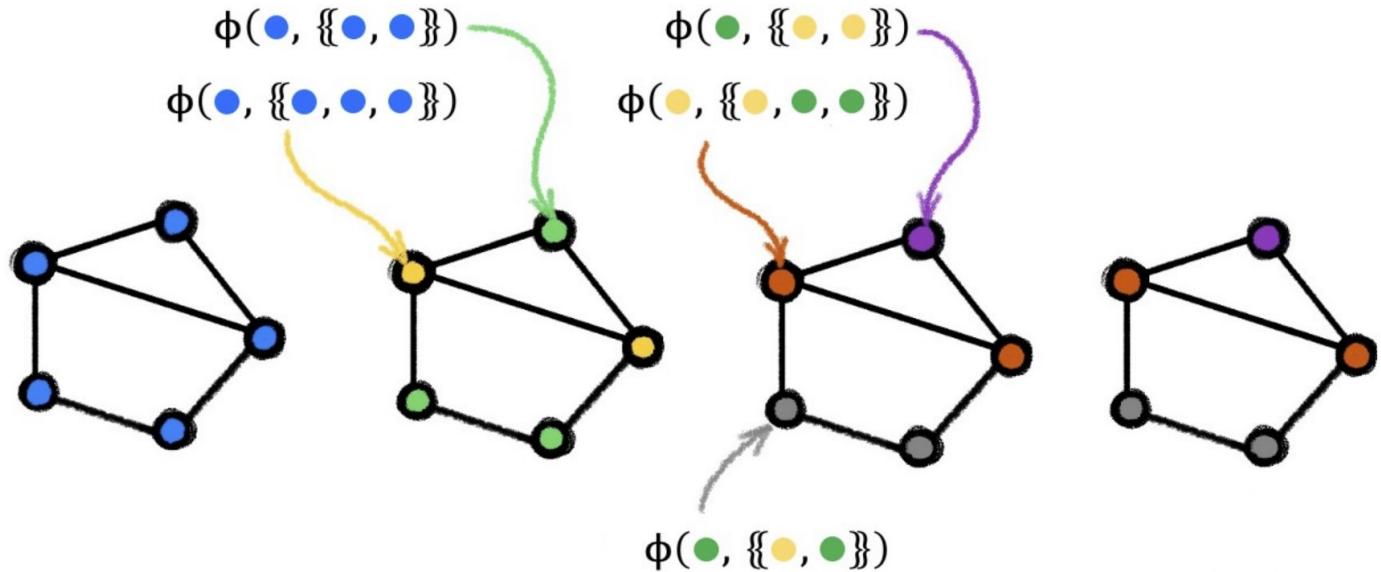
1-WL (Weisfeiler-Lehman) test



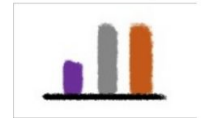
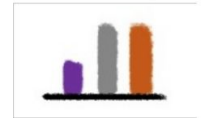
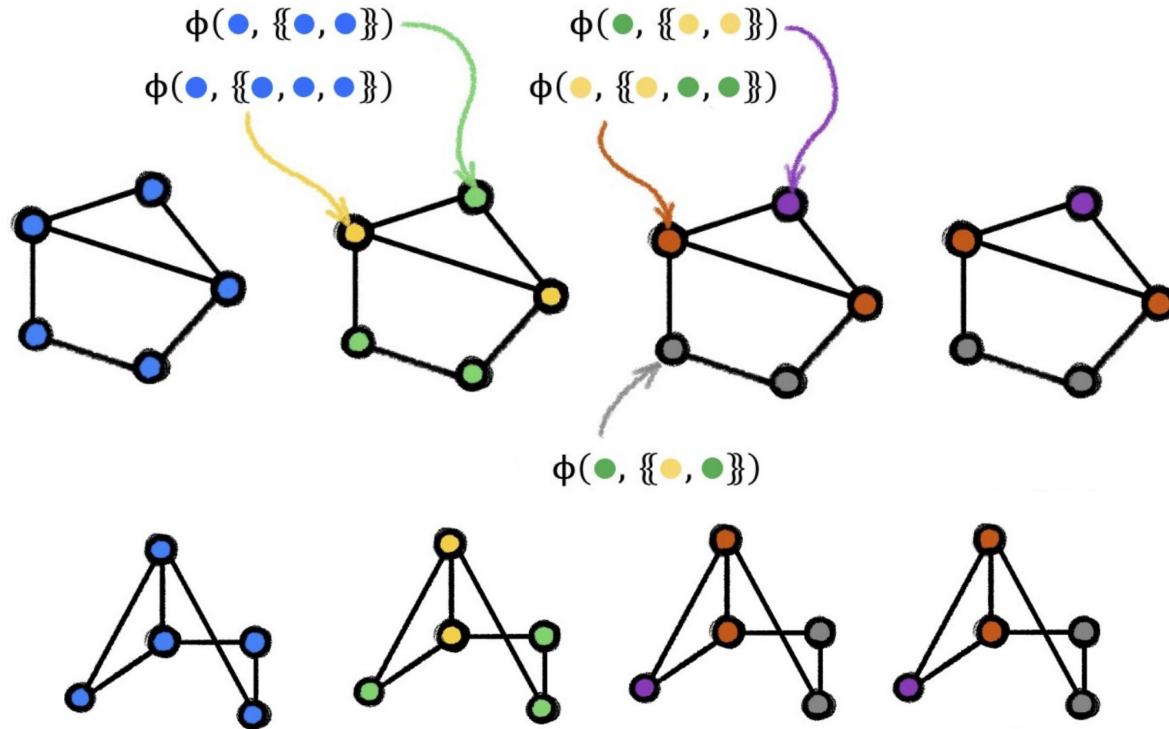
1-WL (Weisfeiler-Lehman) test



1-WL (Weisfeiler-Lehman) test

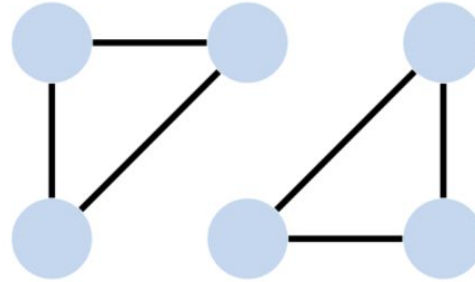
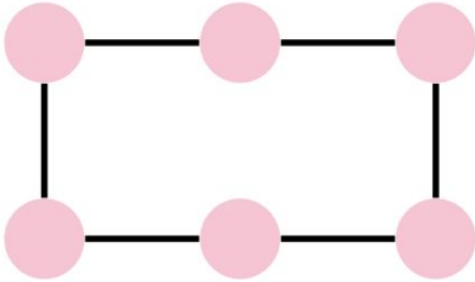


1-WL (Weisfeiler-Lehman) test



1-WL test failures

- The test cannot distinguish these two graphs



Expressive power of GNNs

- GNNs over discrete features can be as only as expressive as 1-WL

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

Input: Initial node coloring $(h_1^{(0)}, h_2^{(0)}, \dots, h_N^{(0)})$

Output: Final node coloring $(h_1^{(T)}, h_2^{(T)}, \dots, h_N^{(T)})$

$t \leftarrow 0$;

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*;

Expressive power of GNNs

- GNNs over discrete features can be as only as expressive as 1-WL
- There is no guarantee that we even reach 1-WL test performance
- Graph Isomorphic Network propose maximally expressive GNNs

$$\mathbf{h}_u = \phi \left((1 + \epsilon)\mathbf{h}_u + \sum_{v \in \mathcal{N}_v} \mathbf{h}_v \right)$$

Expressive power of GNNs

- GNNs over discrete features can be as only as expressive as 1-WL
- There is no guarantee that we even reach 1-WL test performance
- Graph Isomorphic Network propose maximally expressive GNNs
- Since 1-WL test has failure modes -> GNNs inherit these problems

Expressive power of GNNs

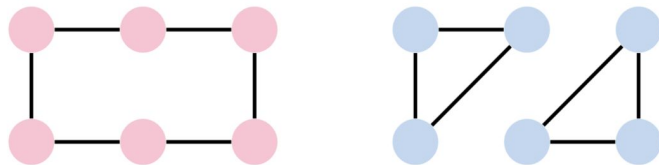
- GNNs over discrete features can be as only as expressive as 1-WL
- There is no guarantee that we even reach 1-WL test performance
- Graph Isomorphic Network propose maximally expressive GNNs
- Since 1-WL test has failure modes -> GNNs inherit these problems

Remedy: input to the GNN information, it is not able to compute

Expressive power of GNNs

- GNNs over discrete features can be as only as expressive as 1-WL
- There is no guarantee that we even reach 1-WL test performance
- Graph Isomorphic Network propose maximally expressive GNNs
- Since 1-WL test has failure modes -> GNNs inherit these problems

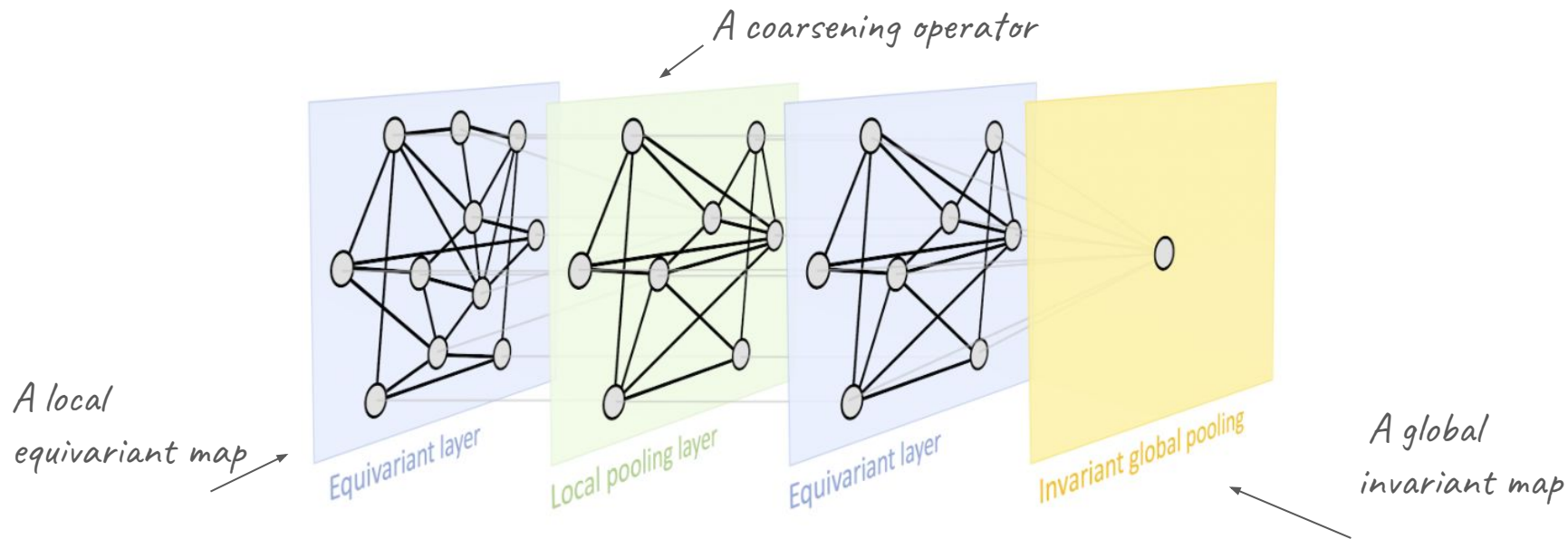
How many clusters are in the graph?



What is the largest cycle?

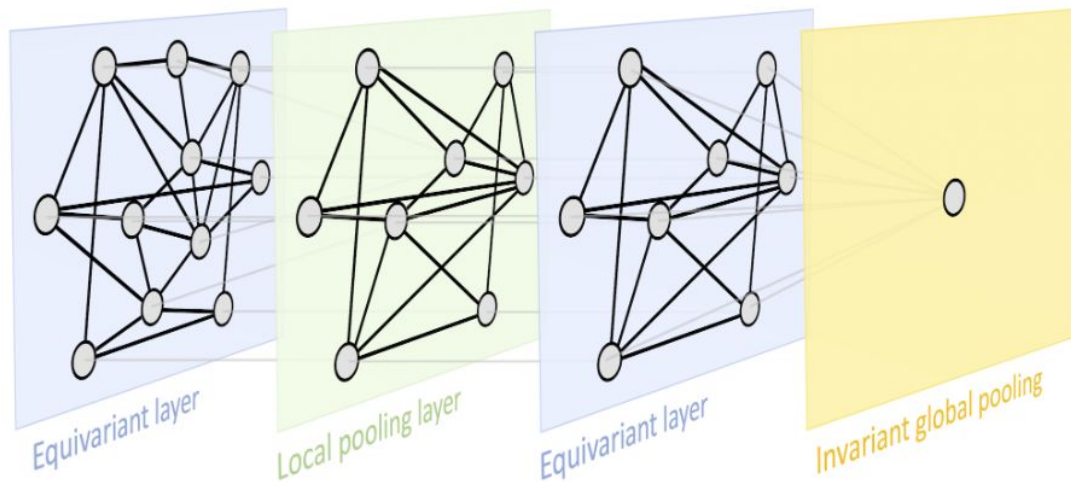
Connecting back to GDL

Geometric Deep Learning Blueprint



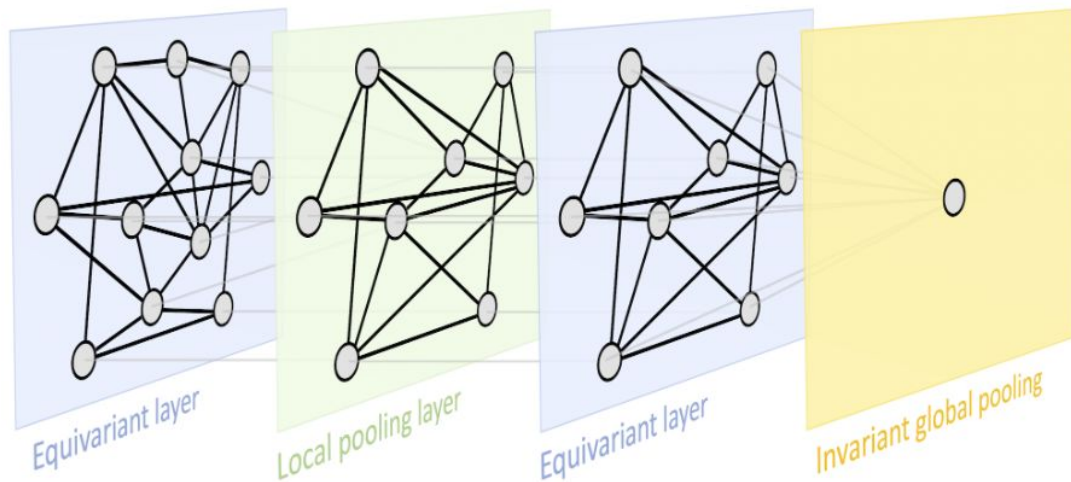
Geometric Deep Learning Blueprint: CNNs

- Convolutional layers equivariant to **translations**
- Local pooling coarsening of the **grid**
- Global pooling (translation invariant) such as mean / sum



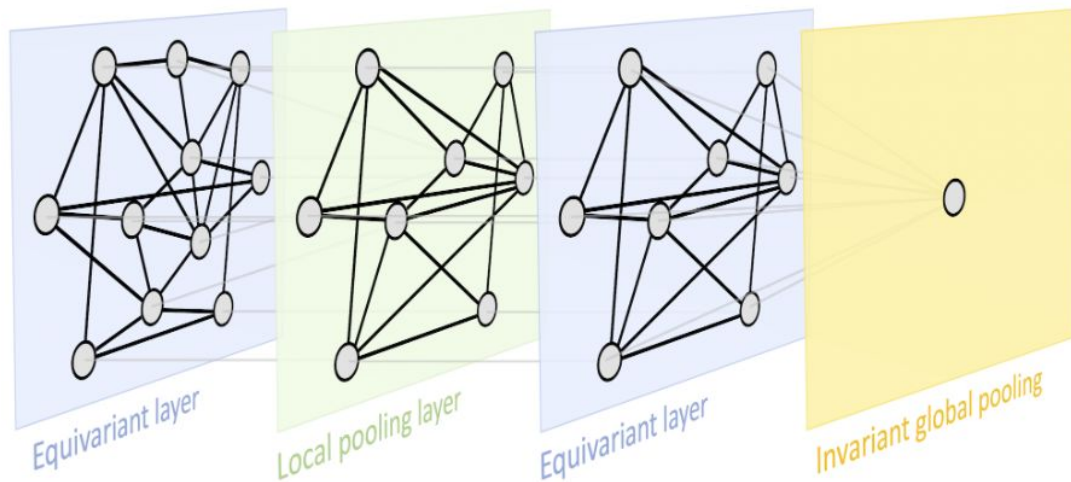
Geometric Deep Learning Blueprint: GNNs

- **Permutation**-equivariant GNN layers
- Local pooling: coarsening of the **graph**
- Global pooling such as mean / sum



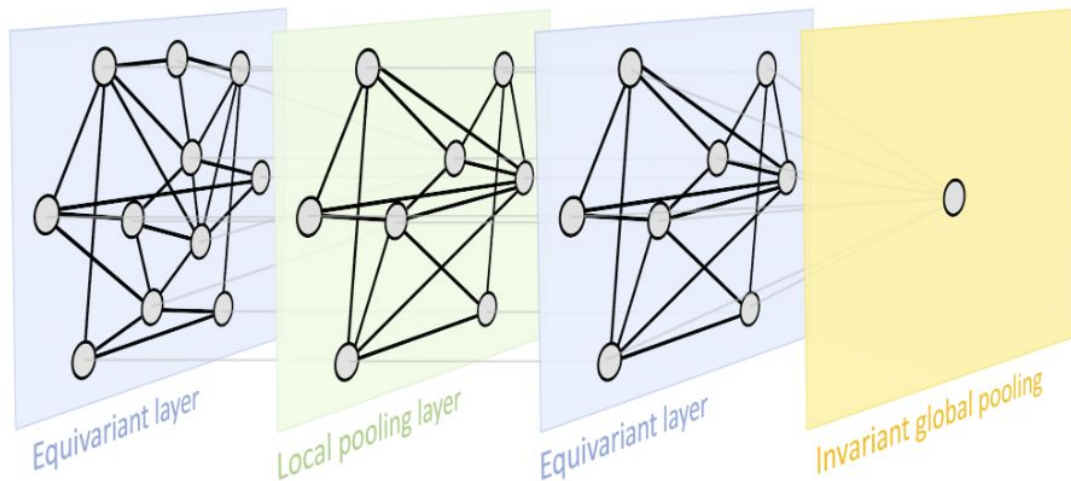
Geometric Deep Learning Blueprint: beyond

- Equivariant layers wrt an **arbitrary group**
- Local pooling: coarsening of the **domain**
- Global pooling such as mean / sum



Geometric Deep Learning Blueprint: beyond

- Equivariant layers wrt an **arbitrary group**
- Local pooling: coarsening of the **domain**
- Global pooling such as mean / sum



Next time!

Summary

Lecture recap

- Graphs and invariance / equivariance to permutations
 - Graph terminology
 - Motivating examples for graph applications
 - Desired properties of neural networks for graph-structured data

Lecture recap

- Graphs and invariance / equivariance to permutations
 - Graph terminology
 - Motivating examples for graph applications
 - Desired properties of neural networks for graph-structured data
- Graph Neural Networks
 - Special cases
 - Main building blocks
 - Expressivity

Lecture recap

- Graphs and invariance / equivariance to permutations
 - Graph terminology
 - Motivating examples for graph applications
 - Desired properties of neural networks for graph-structured data
- Graph Neural Networks
 - Special cases
 - Main building block
 - Expressivity
- Geometric Deep Learning Blueprint

Exercise recap

- Calculate graph Laplacian, explain the connection with GCN
- Check if two graphs are isomorphic
- Small data exploration task
- Step-by-step implementation of Graph Attentional Layer for GAT
- Optional implementation of GCN layer (optional, for comparison)

Recommended reading

- Graph representation Learning Book: especially Ch. 2.3 and Ch.7.
- Geometric Deep Learning Book: Ch. 3.5, Ch. 4.1, Ch. 5