AdTrans project, 2023-1-PL01-KA220-HED-000158917

ADTRANS
ADVANCED DATA ANALYSIS FOR TRANSPORTATION AND MOBILITY ACADEMIC TEACHERS

Session 3

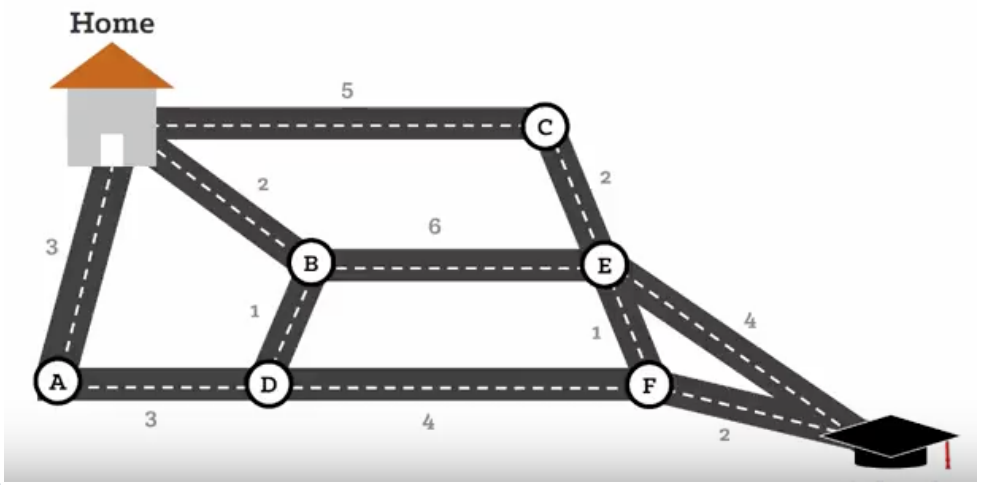
Graph Convolutional Neural Nets

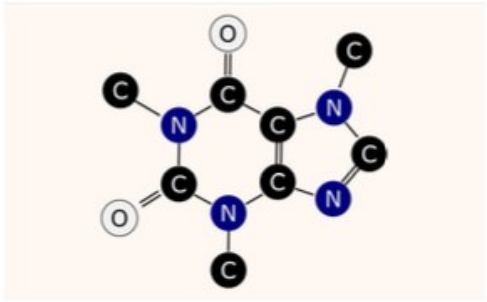
Paraskevi Fasouli

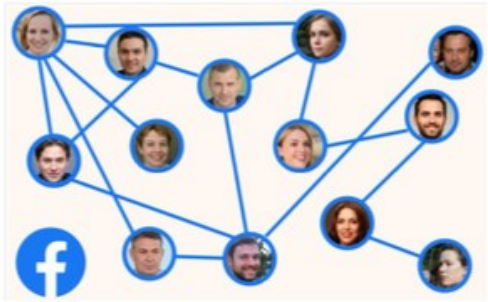
Co-funded by the European Union

Graph Usage

- Crucial role in intelligent transportation systems
- Ability to model complex relationships and interactions within transportation networks







Importance of Graphs

1. Representation of Transportation Networks

- Graphs represent transportation networks where nodes can signify intersections, stations, or stops, and edges can represent roads, railways, or routes.

2. Efficient Route Planning and Navigation

- Shortest Path Algorithms: Algorithms like Dijkstra's and A* can be applied to graph representations of road networks to find the shortest or fastest routes between points.
- Real-Time Traffic Management: Graphs enable the integration of real-time traffic data, allowing for dynamic route adjustments and optimization.

3. Traffic Flow Analysis

- Flow Models: Traffic flow can be modeled using graph-based approaches to understand and predict traffic congestion patterns.
- Bottleneck Identification: Graph analysis can help identify critical points or bottlenecks in the transportation network that cause delays and congestion.

4. Public Transportation Optimization

- Timetable Scheduling: Graphs can model public transportation systems, aiding in the scheduling and synchronization of buses, trains, and other modes of transport.
- Passenger Flow: to optimize routes and to minimize wait times and overcrowding.

5. Infrastructure Planning and Management

- Network Design: Graph theory aids in the design and expansion of transportation infrastructure by identifying optimal locations for new roads, bridges, and transit lines.

6. Safety and Incident Management

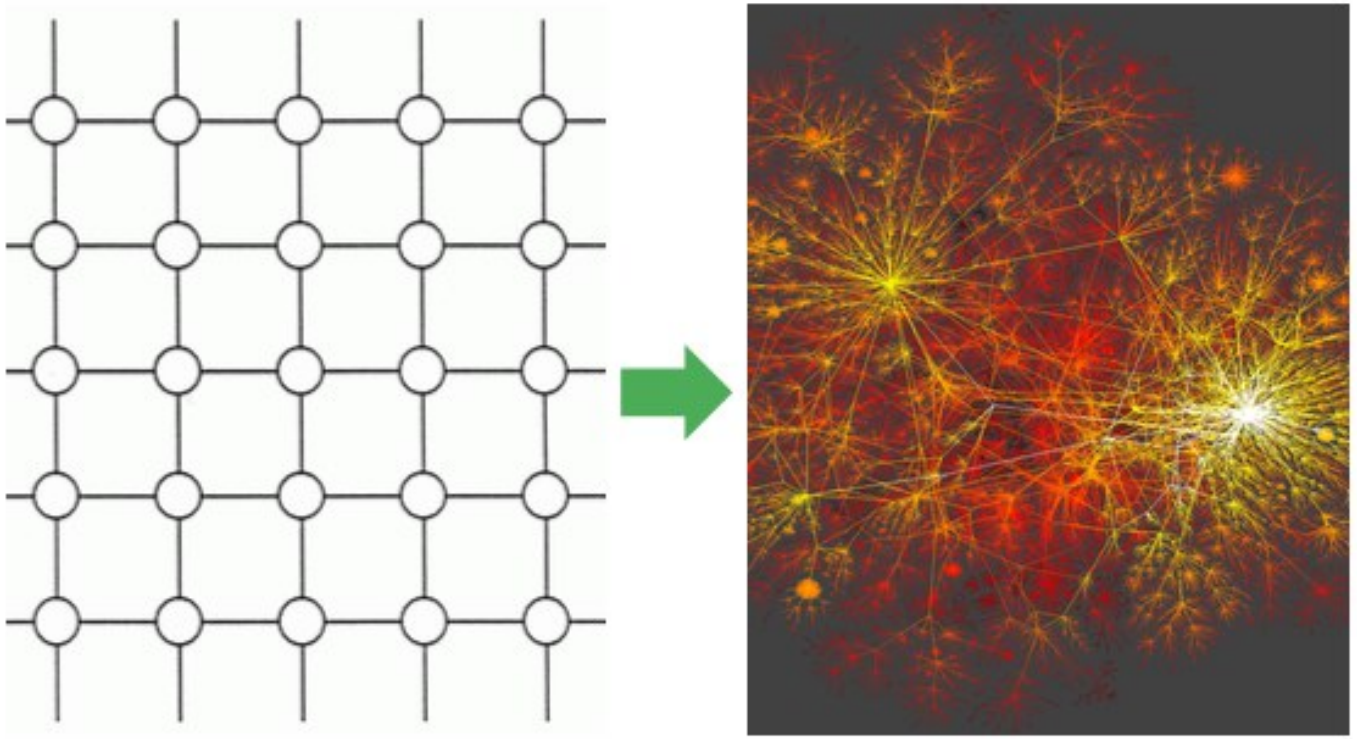
- Incident Detection: Graphs can be used to model and analyze the propagation of incidents or hazards through a transportation network.
- Emergency Response: Efficient routing for emergency vehicles can be determined using graph-based methods, ensuring quick response times.

7. Data Integration and Multimodal Transportation

- Multimodal Networks: Graphs can integrate various modes of transportation (e.g., buses, trains, bikes) into a unified framework for seamless trip planning.

From CNN to Graph CNN

- ✔ Graph CNNs are a type of neural network specifically designed to operate on graph-structured data.
- ✔ Convolution is well defined in Euclidean data, grid-like data e.g. images
- ✔ Not straightforward to define convolution on irregular network, widely observed in real world



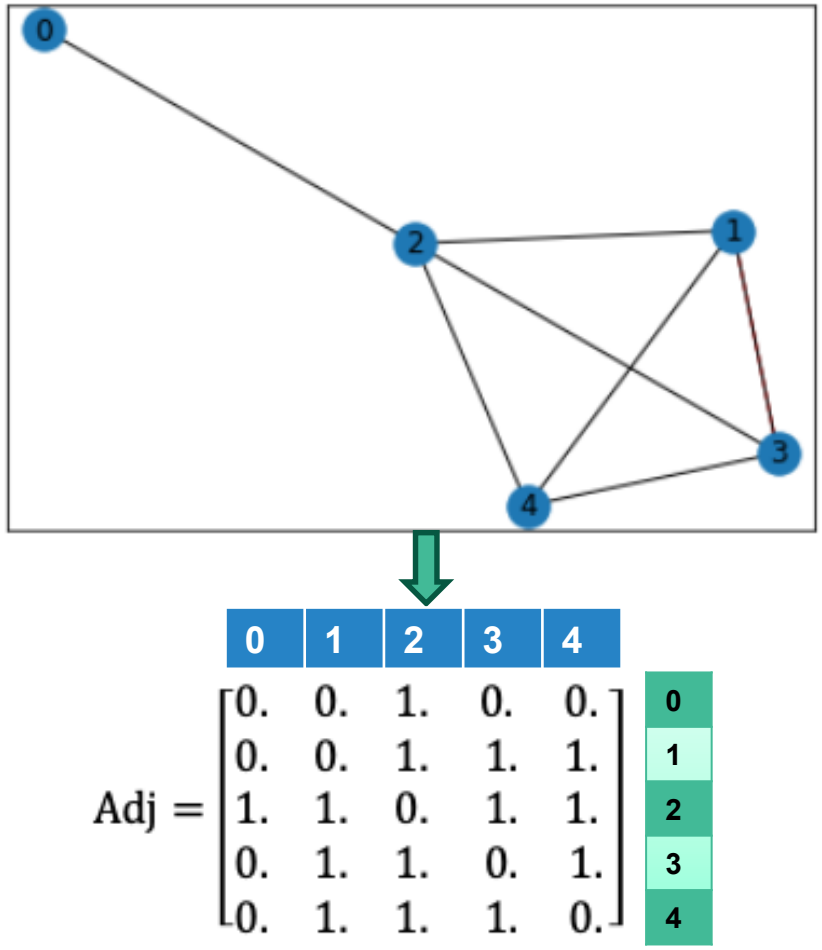
Grid-like network

Irregular networks

Graphs & Their Representations

- ✔ **Nodes and Edges**: A graph consists of nodes (vertices) and edges (links) connecting them. Each node can represent an entity, and each edge represents a relationship or interaction between entities.

- ✔ **Adjacency Matrix**: A common way to represent a graph is through an adjacency matrix A, where Aij indicates the presence (and possibly weight) of an edge between nodes i and j.



Tasks of GNNs

1. Denote a graph with $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{W}, \boldsymbol{X})$
   - $\boldsymbol{V}$ is the node set with $\boldsymbol{n} = |\boldsymbol{V}|$, $\boldsymbol{E}$ is the edge set, and $\boldsymbol{W} \in \boldsymbol{R}^{n \times n}$ is the weighted adjacency matrix
   - Each node is associated with $\boldsymbol{d}$ features, and $\boldsymbol{X} \in \boldsymbol{R}^{n \times d}$ is the feature matrix of nodes
2. Perform a task (e.g Node Classification)

Node Classification:

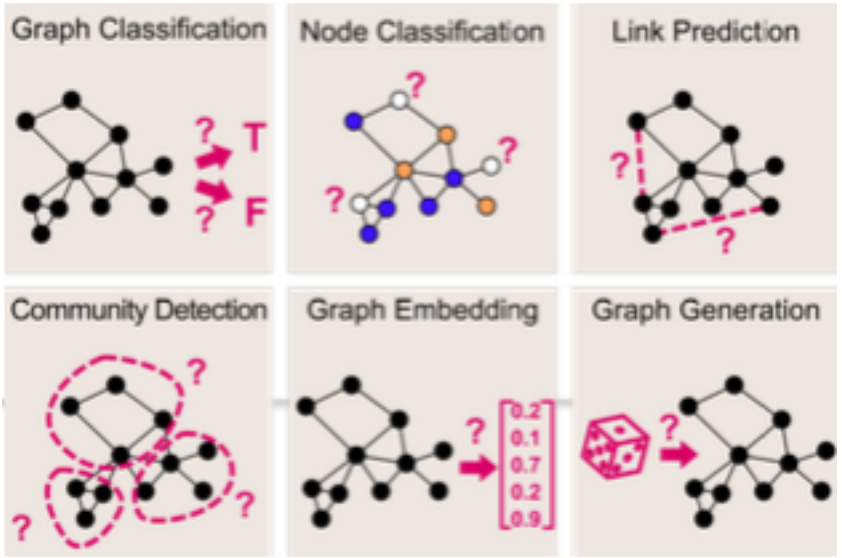
- Predicting the label of each node in the graph, such as classifying users in a social network or proteins in a biological network.

Link Prediction:

- Predicting the existence of an edge between two nodes, useful in recommendation systems and knowledge graph completion.

Graph Classification:

- Classifying entire graphs, such as predicting molecular properties in chemistry or classifying documents in natural language processing.



Existing methods to define convolution

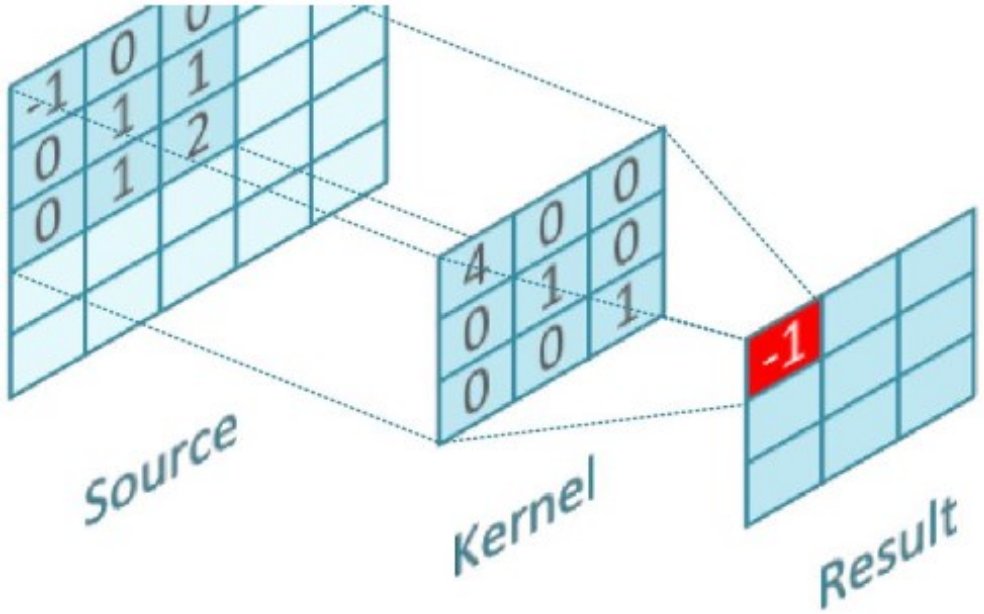
Spectral methods: define convolution in spectral domain

- Convolution is defined via graph **Fourier transform** and **convolution theorem**.
- The main challenge is that convolution filter defined in spectral domain is not localized in vertex domain.

Spatial methods: define convolution in the vertex domain

- Convolution is defined as a weighted average function over all vertices located **in the neighborhood of** target vertex.
- The main challenge is that the size of neighbourhood varies remarkably across nodes, e.g., power-law degree distribution.

Normal Convolution



Result = (-1 x 4) + (0 x 0) + (0 x 0) + (0 x 0) + (1 x 1) + (1 x 0) + (0 x 1) + (1 x 0) + (2 x 1) = -1

Aggregates information from neighbouring pixels

Graph Convolution

Idea: Aggregate information from a node's neighbors to update its representation.

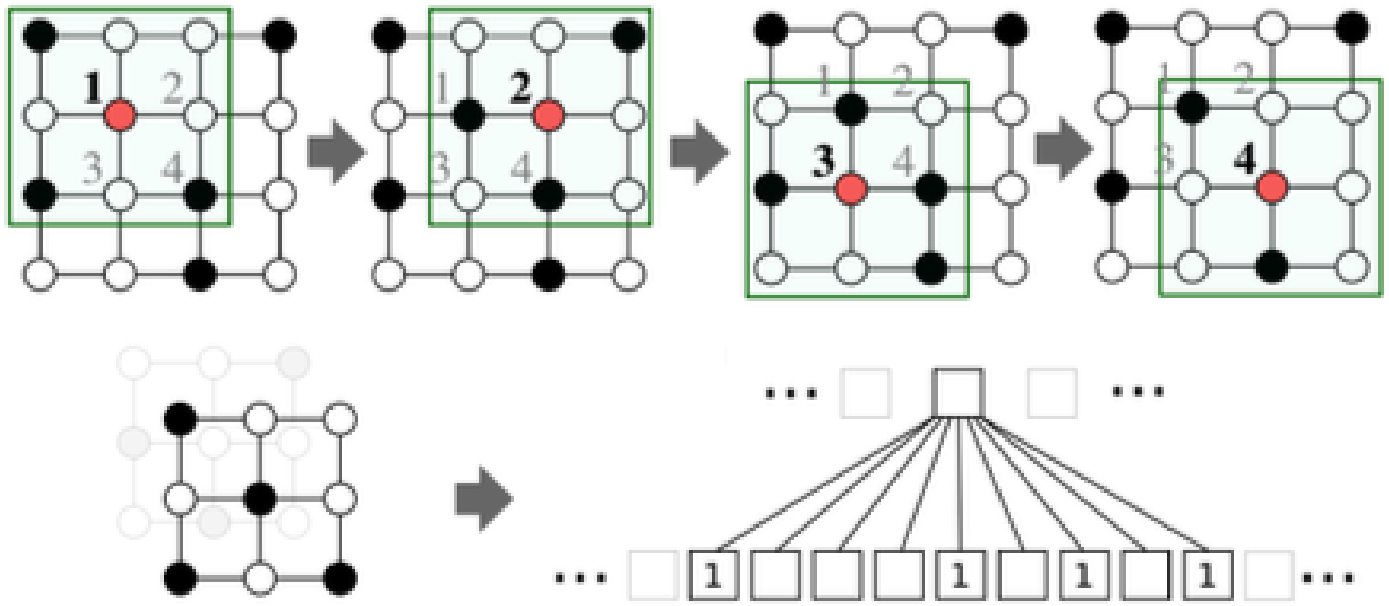
1. Neighborhood Aggregation:

- For a given node, the graph convolution operation aggregates information from its neighboring nodes. This is typically done through a weighted sum of the features of the node and its neighbors.

2. Feature Transformation:

- After aggregation, the combined information is passed through a transformation, usually a linear transformation followed by a non-linear activation function.

Graph Convolution



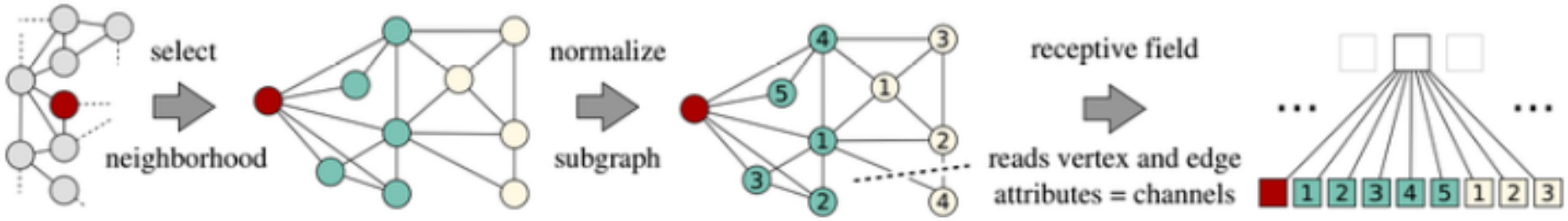
1. Determine Neighbourhood

2. Impose an order in neighbourhood

3. Share parameters

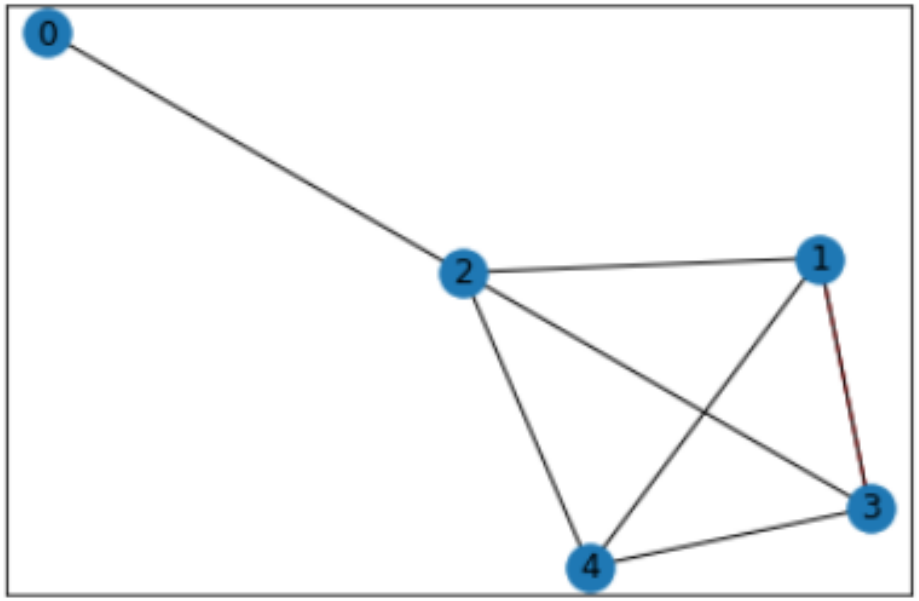
Graph Convolution

- ✔ For each node, select the fixed number of nodes as its neighbouring nodes, according to certain proximity metric
- ✔ Impose an order according to the proximity metric



1. Determine Neighbourhood → 2. Impose an order in neighbourhood → 3. Share parameters

Graph Convolution



$$\text{Adj} = \begin{bmatrix} 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 1. & 1. & 1. \\ 1. & 1. & 0. & 1. & 1. \\ 0. & 1. & 1. & 0. & 1. \\ 0. & 1. & 1. & 1. & 0. \end{bmatrix}, X = \begin{bmatrix} 1 & -1 \\ 2 & -2 \\ 1 & -1 \\ 3 & -3 \\ 4 & -4 \end{bmatrix}$$

$\boldsymbol{X} \in \boldsymbol{R}^{n \times d}$ is the feature matrix of the nodes

Graph Convolution

$$A = \begin{bmatrix} 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 1. & 1. & 1. \\ 1. & 1. & 0. & 1. & 1. \\ 0. & 1. & 1. & 0. & 1. \\ 0. & 1. & 1. & 1. & 0. \end{bmatrix}, X = \begin{bmatrix} 1 & -1 \\ 2 & -2 \\ 1 & -1 \\ 3 & -3 \\ 4 & -4 \end{bmatrix}$$

$$AX = \begin{bmatrix} 1 & -1 \\ 8 & -8 \\ 10 & -10 \\ 7 & -7 \\ 6 & -6 \end{bmatrix}$$

$\boldsymbol{X} \in \boldsymbol{R}^{n \times d}$ is the feature matrix of the nodes

Two Problems :

1. The aggregate feature does not contain itself features
2. The node with large degree has high values

Graph Convolution

Solution to the 1st problem: Adding self-loops (I) to adjacency matrix (A)

$$A + I = \hat{A} = \begin{bmatrix} 1. & 0. & 1. & 0. & 0. \\ 0. & 1. & 1. & 1. & 1. \\ 1. & 1. & 1. & 1. & 1. \\ 0. & 1. & 1. & 1. & 1. \\ 0. & 1. & 1. & 1. & 1. \end{bmatrix}, X = \begin{bmatrix} 1 & -1 \\ 2 & -2 \\ 1 & -1 \\ 3 & -3 \\ 4 & -4 \end{bmatrix} \quad (A + I)X = \hat{A}X \quad AX = \begin{bmatrix} 2 & -2 \\ 10 & -10 \\ 11 & -11 \\ 10 & -10 \\ 10 & -10 \end{bmatrix}$$

$\boldsymbol{X} \in \boldsymbol{R}^{n \times d}$ is the feature matrix of the nodes

Graph Convolution

Solution to the 2nd problem: Normalizing the Feature Representations

The feature representations can be normalized by node degree by transforming the adjacency matrix (A) by multiplying it with the inverse degree matrix (D)

$$A = \begin{bmatrix} 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 1. & 1. & 1. \\ 1. & 1. & 0. & 1. & 1. \\ 0. & 1. & 1. & 0. & 1. \\ 0. & 1. & 1. & 1. & 0. \end{bmatrix}, X = \begin{bmatrix} 1 & -1 \\ 2 & -2 \\ 1 & -1 \\ 3 & -3 \\ 4 & -4 \end{bmatrix} \Rightarrow D = \begin{bmatrix} 1. & 0. & 0. & 0. & 0. \\ 0. & 3. & 0. & 0. & 0. \\ 0. & 0. & 4. & 0. & 0. \\ 0. & 0. & 0. & 3. & 0. \\ 0. & 0. & 0. & 0. & 3. \end{bmatrix} \xrightarrow{D^{-1}AX} D^{-1}AX = \begin{bmatrix} 1 & -1 \\ 2.66 & -2.66 \\ 2.5 & -2.5 \\ 2.3 & -2.3 \\ 2 & -2 \end{bmatrix}$$

$\boldsymbol{X} \in \boldsymbol{R}^{\boldsymbol{n \times d}}$ is the feature matrix of the nodes

Graph Convolution

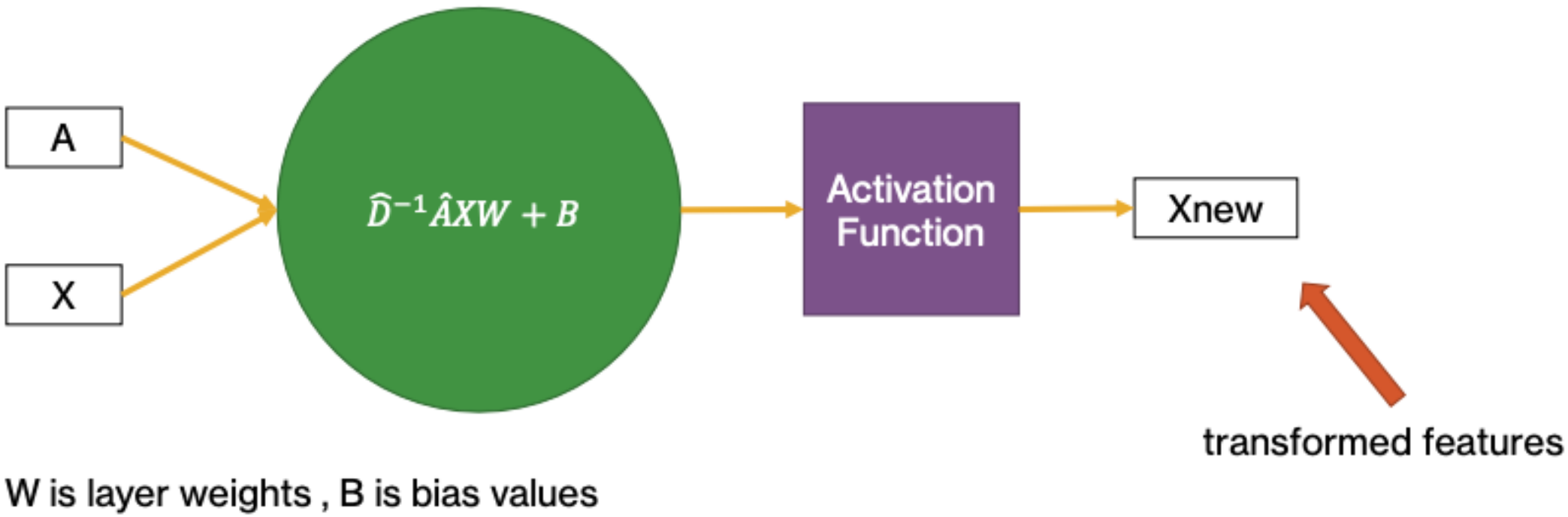
Combine the two techniques to solve both problems:

$$A + I = \hat{A}$$
$$\hat{D}^{-1}\hat{A}X$$

$$A = \begin{bmatrix} 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 1. & 1. & 1. \\ 1. & 1. & 0. & 1. & 1. \\ 0. & 1. & 1. & 0. & 1. \\ 0. & 1. & 1. & 1. & 0. \end{bmatrix}, X = \begin{bmatrix} 1 & -1 \\ 2 & -2 \\ 1 & -1 \\ 3 & -3 \\ 4 & -4 \end{bmatrix} \Rightarrow \hat{D} = \begin{bmatrix} 2. & 0. & 0. & 0. & 0. \\ 0. & 4. & 0. & 0. & 0. \\ 0. & 0. & 5. & 0. & 0. \\ 0. & 0. & 0. & 4. & 0. \\ 0. & 0. & 0. & 0. & 4. \end{bmatrix} \Rightarrow \hat{D}^{-1}\hat{A}X \Rightarrow \hat{D}^{-1}\hat{A}X = \begin{bmatrix} 1 & -1 \\ 2.5 & -2.5 \\ 2.2 & -2.2 \\ 2.5 & -2.5 \\ 2.5 & -2.5 \end{bmatrix}$$

$\boldsymbol{X} \in \boldsymbol{R}^{\boldsymbol{n \times d}}$ is the feature matrix of the nodes

Graph Convolution Layer

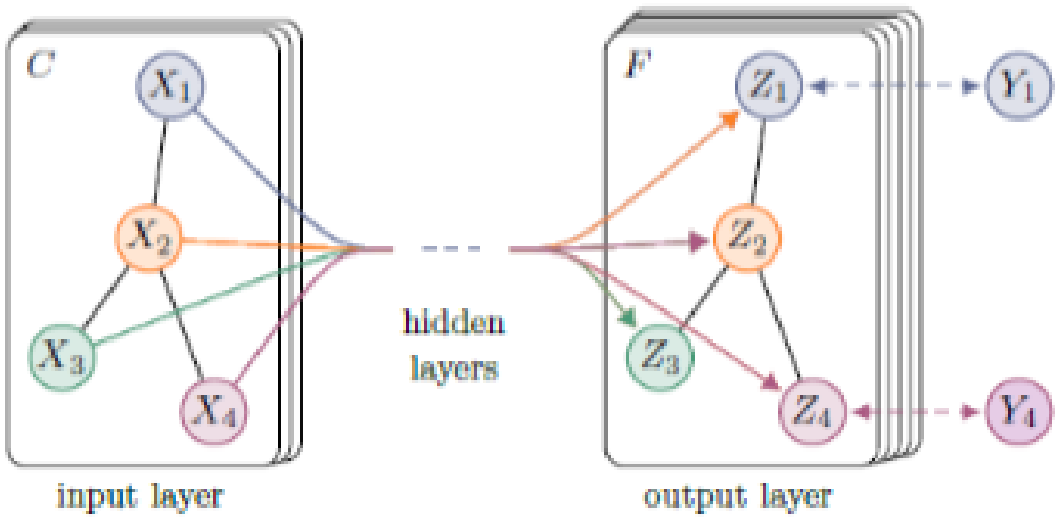


W is layer weights , B is bias values

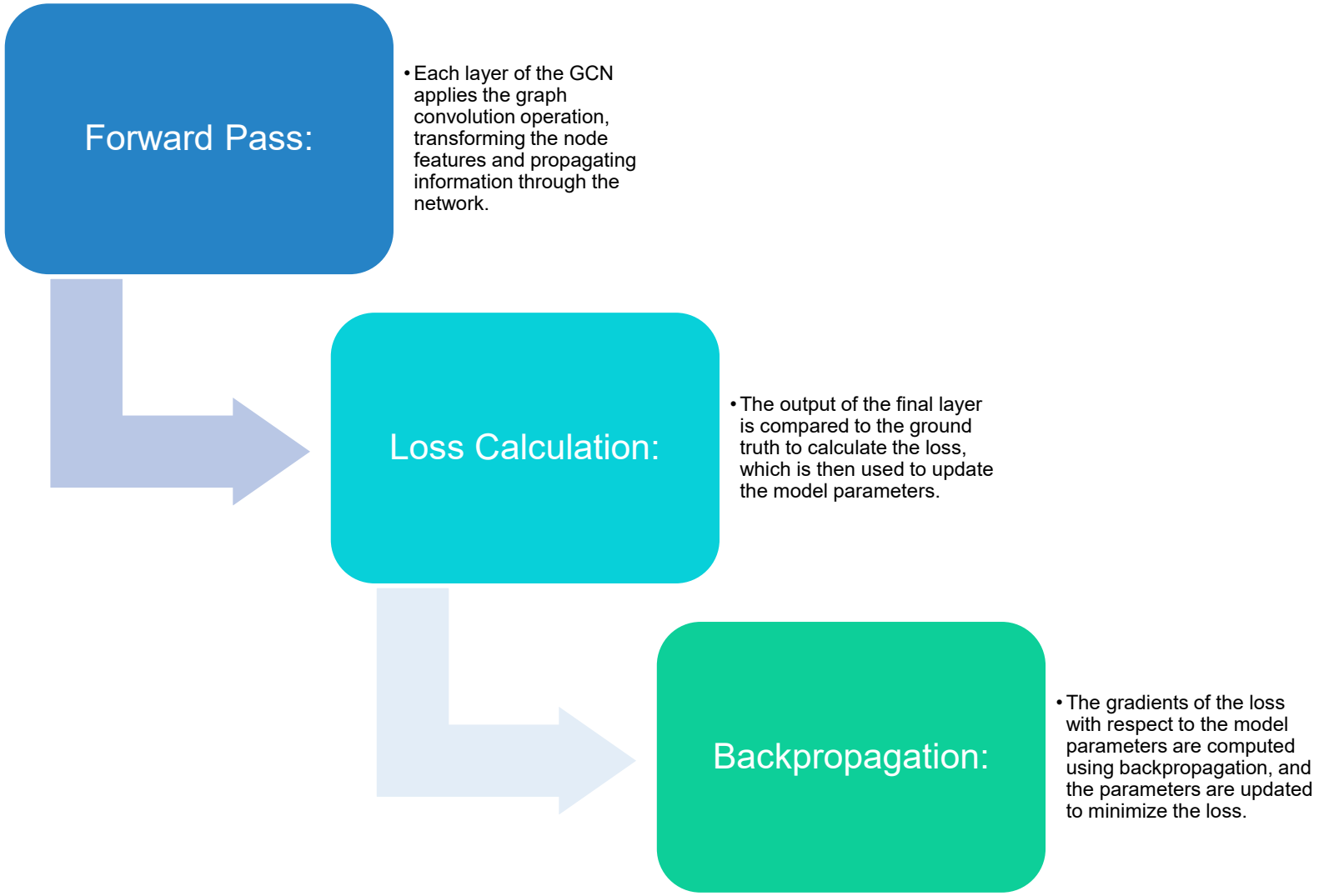
Graph Convolution Layer

$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}X\boxed{W^{(0)}}\Big)\boxed{W^{(1)}}\Big)$$

Parameter for feature Transformation



Learning Process



End of Session3