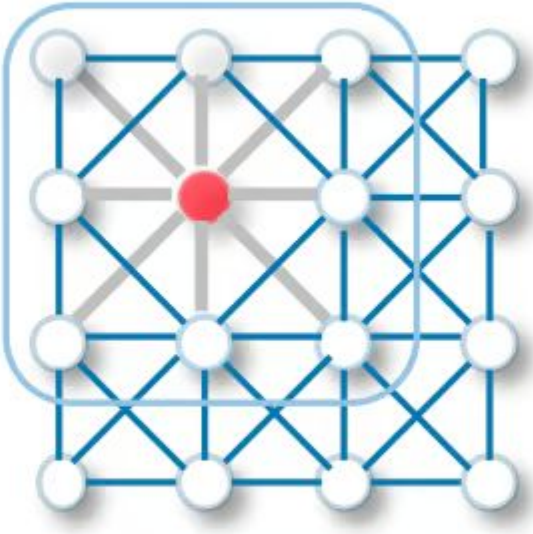
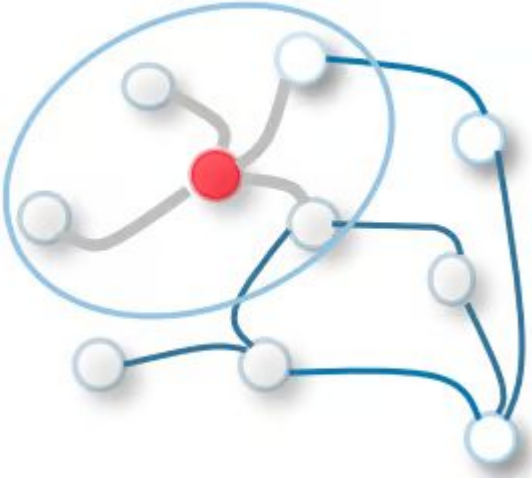
Graph Convolutional Networks (GCN)





Neighborhood Group Chat

Imagine every node (person) in a network participates in a group chat with its neighbors:

- Every node has some information about itself (features).
- In each "round" (layer), a node updates its information by listening to its neighbors and mixing their info with its own. This mixing is called message passing.
- Do it again and again — soon, each node's info reflects not just itself, but its wider community's structure.

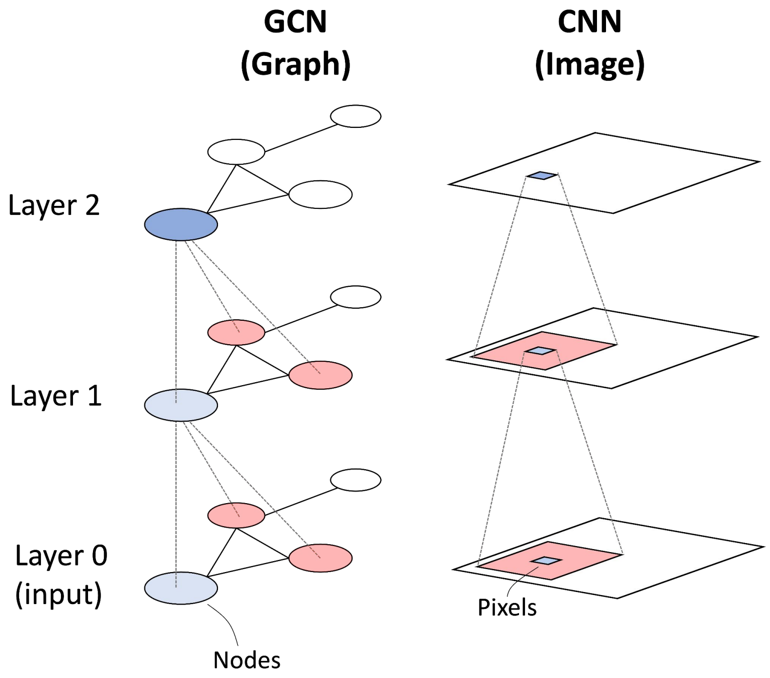
How Does a GCN Work?

Graph Input: Think of your dataset as a web of nodes and edges. Nodes have features (e.g. name, age, molecule type).

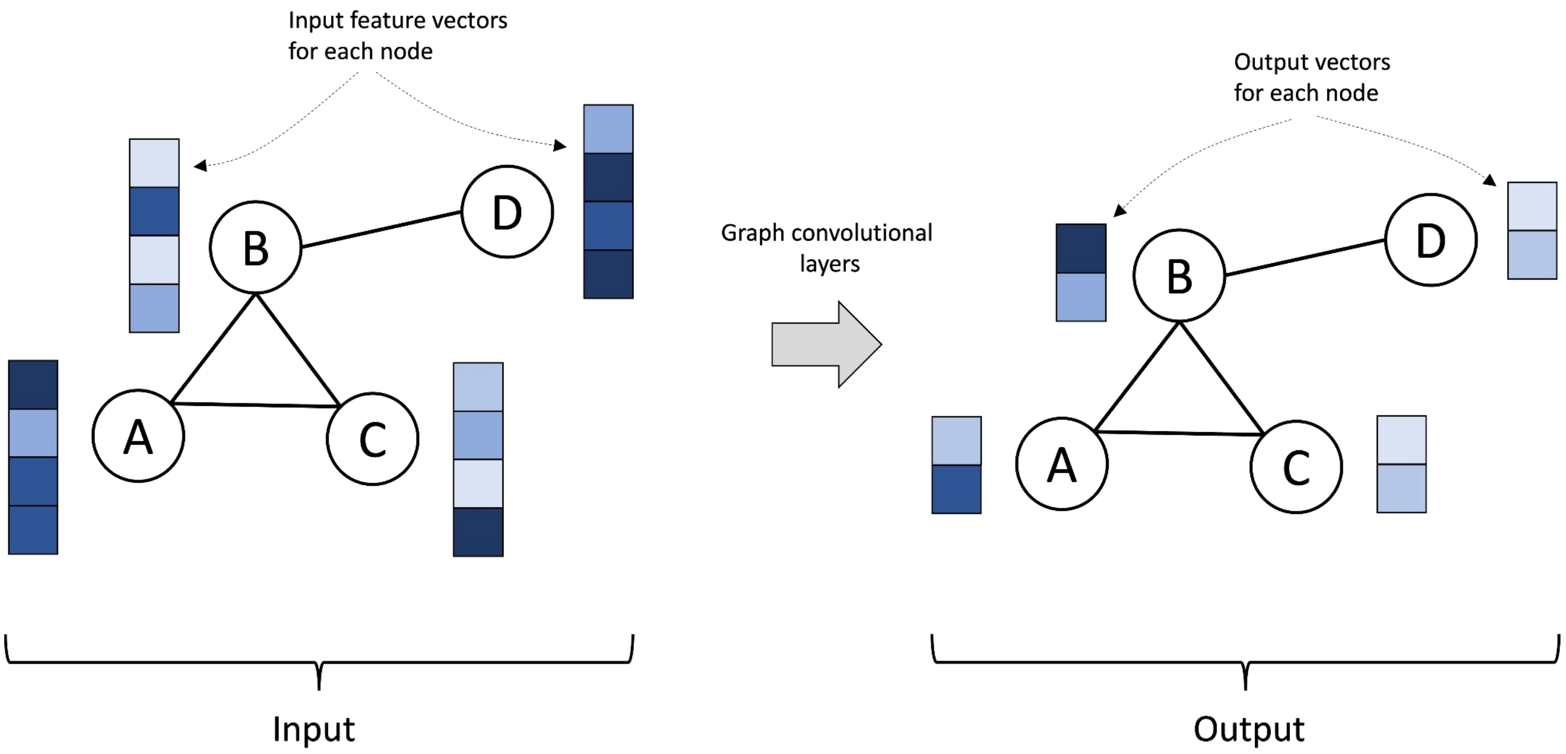
Convolution (Message Passing): Each node gathers info from its neighbors, merges it with its own, then runs it through a learnable function (like mixing ingredients in a recipe).

Multiple Layers = Wider Reach: More layers mean each node gets information from nodes that are farther away (friend-of-a-friend, then friend-of-a-friend-of-a-friend, etc.).

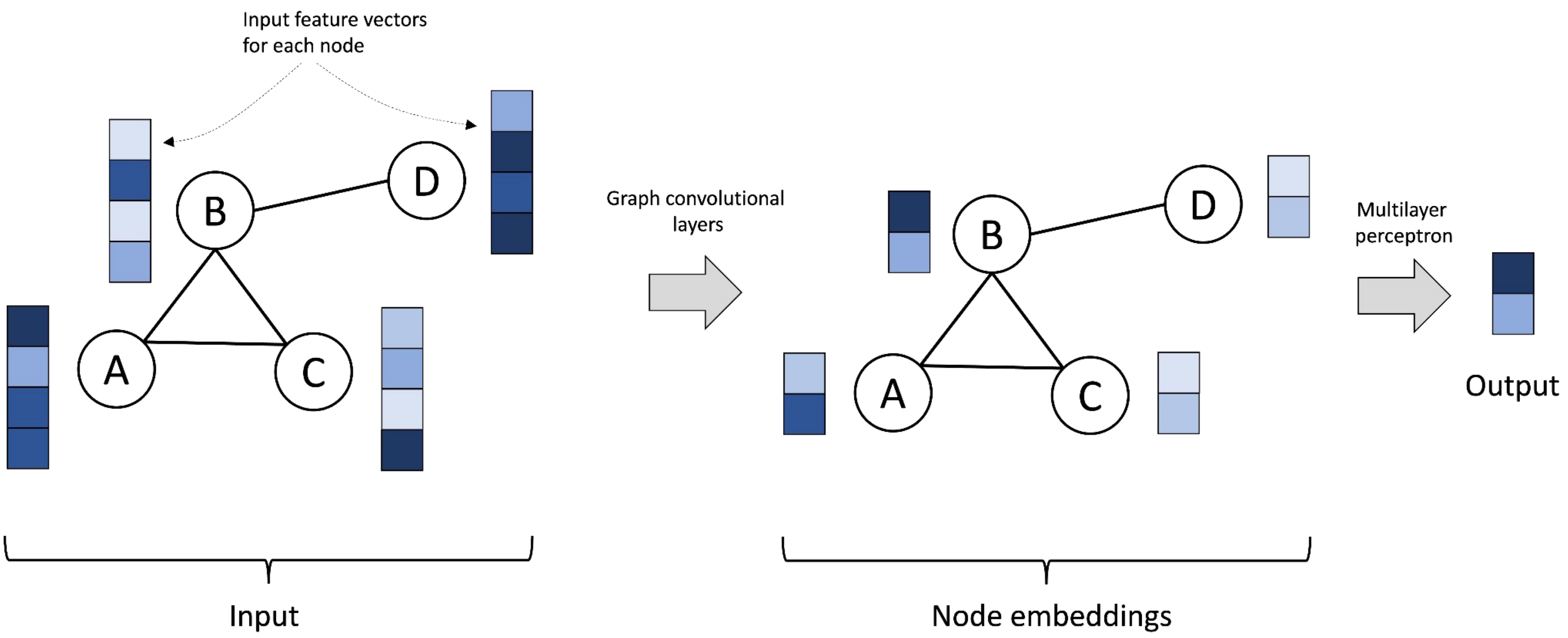
Prediction: After a few layers, nodes' new representations can be used for tasks like classifying nodes, predicting links, understanding the graph.



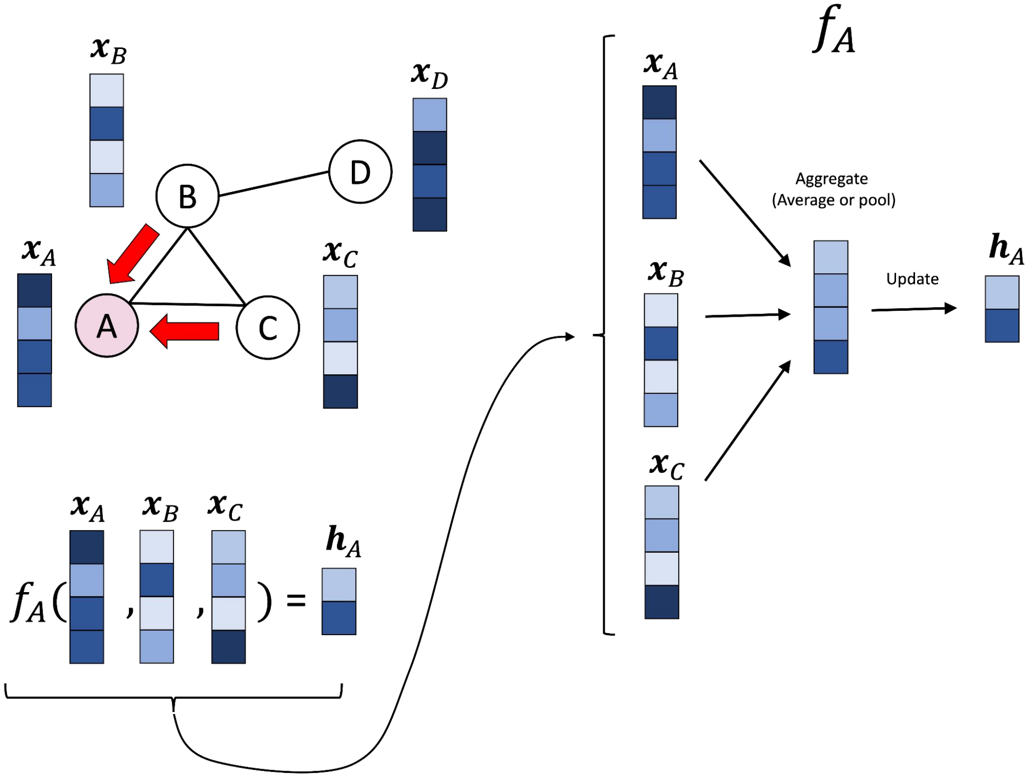
Inputs and outputs of a GCN



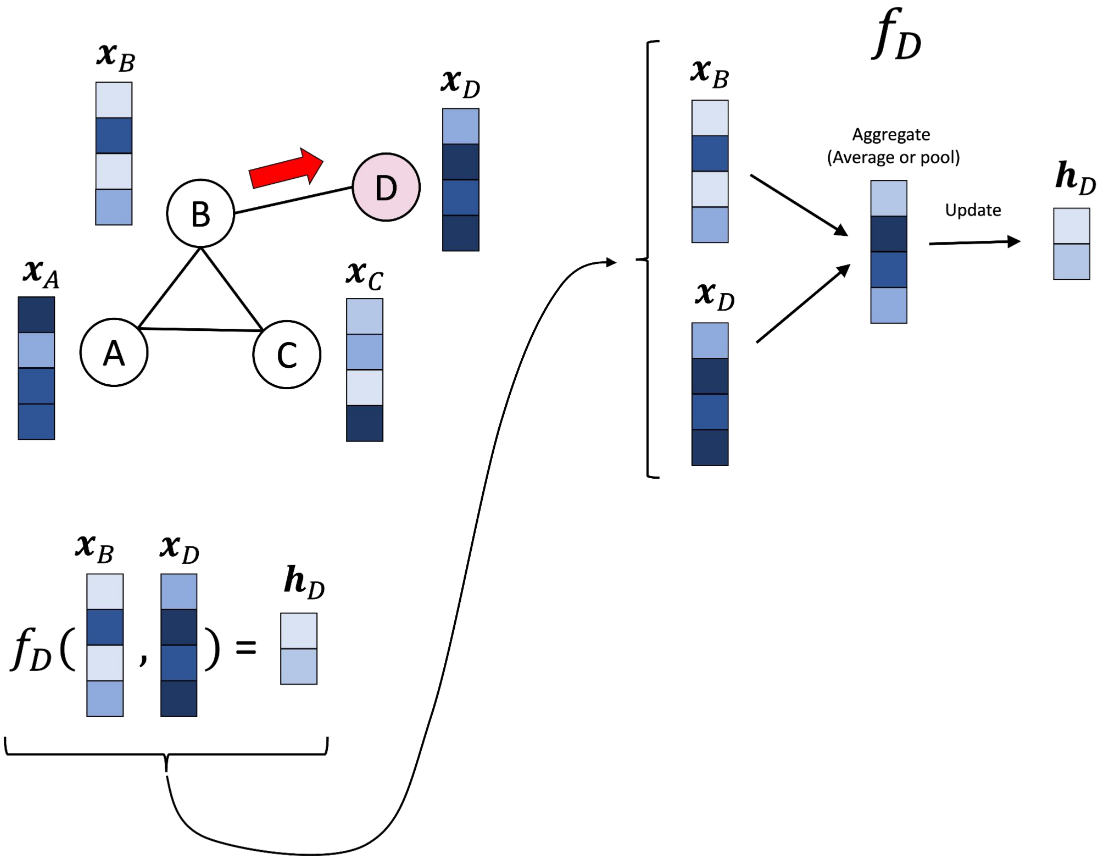
Graph Classification



The Graph Convolutional Layer



The Graph Convolutional Layer



How exactly does a GCN perform the aggregation and update?

$$f(\boldsymbol{X},\boldsymbol{A}) := \sigma\left(\boldsymbol{D}^{-1/2}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-1/2}\boldsymbol{X}\boldsymbol{W}\right)$$

where,

$$\begin{aligned}
\boldsymbol{A} \in \mathbb{R}^{\mathrm{n\times n}} &:= \text{The adjacency matrix} \\
\boldsymbol{I} \in \mathbb{R}^{\mathrm{n\times n}} &:= \text{The identity matrix} \\
\boldsymbol{D} \in \mathbb{R}^{\mathrm{n\times n}} &:= \text{The degree matrix of } \boldsymbol{A}+\boldsymbol{I} \\
\boldsymbol{X} \in \mathbb{R}^{\mathrm{n\times d}} &:= \text{The input data (i.e., the per-node feature vectors)} \\
\boldsymbol{W} \in \mathbb{R}^{\mathrm{d\times w}} &:= \text{The layer's weights} \\
\sigma(.) &:= \text{The activation function (e.g., ReLU)}
\end{aligned}$$

What exactly is the equation?

$$f(\boldsymbol{X},\boldsymbol{A}) \coloneqq \sigma\Big(\underbrace{\underbrace{\underbrace{\boldsymbol{D}^{-1/2}\underbrace{(\boldsymbol{A}+\boldsymbol{I})}_{\text{Add self-loops}}\boldsymbol{D}^{-1/2}}_{\text{Normalize adjacency matrix}}\boldsymbol{X}}_{\text{Aggregate}}\boldsymbol{W}}_{\text{Update}}\Big)$$

Lets Decode

- A + I -> Self Loops
- D is the degree matrix of A+I

$$\boldsymbol{D} := \begin{bmatrix} d_{1,1} & 0 & 0 & \dots & 0 \\ 0 & d_{2,2} & 0 & \dots & 0 \\ 0 & 0 & d_{3,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & d_{n,n} \end{bmatrix}$$

where $d_{i,i}$ is the number of adjacent nodes (i.e., direct neighbors) to node i.

The matrix $\boldsymbol{D}^{-1/2}$ is the matrix formed by taking the reciprocal of the square root of each entry in $\boldsymbol{D}$. That is,

$$\boldsymbol{D}^{-1/2} := \begin{bmatrix} \frac{1}{\sqrt{d_{1,1}}} & 0 & 0 & \dots & 0 \\ 0 & \frac{1}{\sqrt{d_{2,2}}} & 0 & \dots & 0 \\ 0 & 0 & \frac{1}{\sqrt{d_{3,3}}} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{\sqrt{d_{n,n}}} \end{bmatrix}$$

“Normalizing” the adjacency matrix

$$\tilde{A} := D^{-1/2}(A + I)D^{-1/2}$$

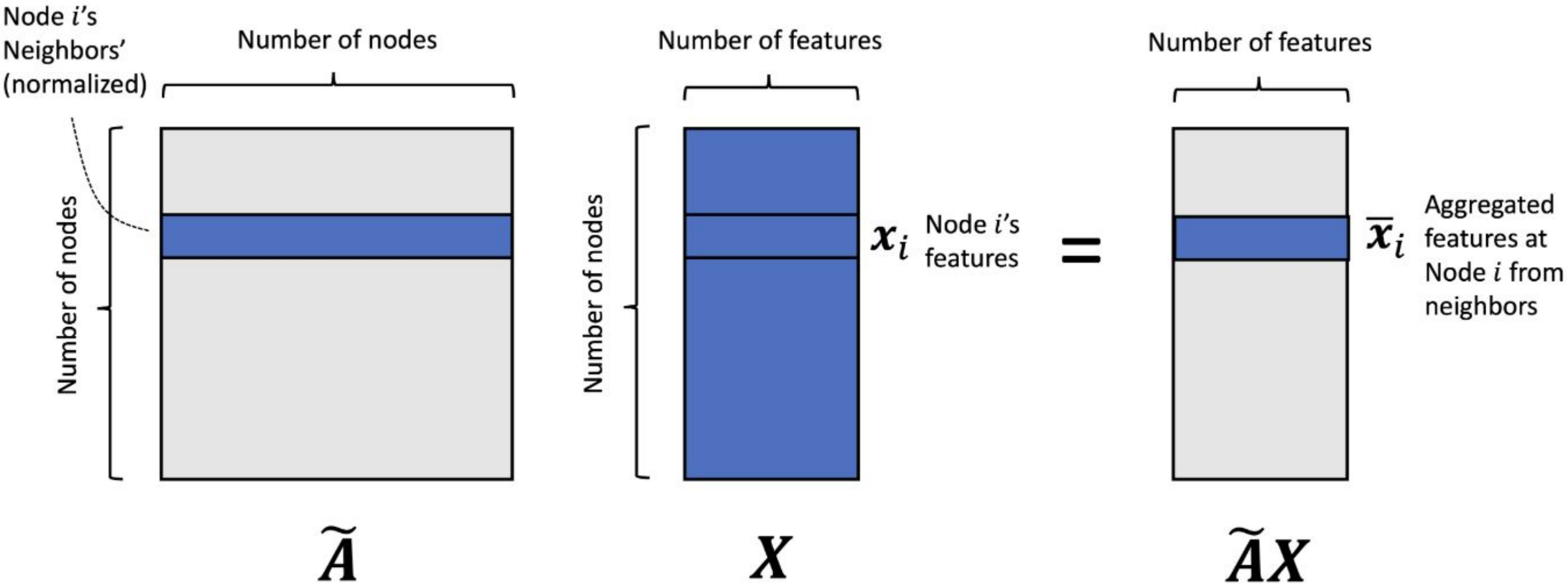
Then,

$$\tilde{A}_{i,j} := \begin{cases} \frac{1}{\sqrt{d_{i,i}d_{j,j}}}, & \text{if there is an edge between node i and j} \\ 0, & \text{otherwise} \end{cases}$$

With this notation, we can simplify the graph convolutional layer function as follows:

$$f(X, A) := \sigma\left(\tilde{A}XW\right)$$

Lets Combine



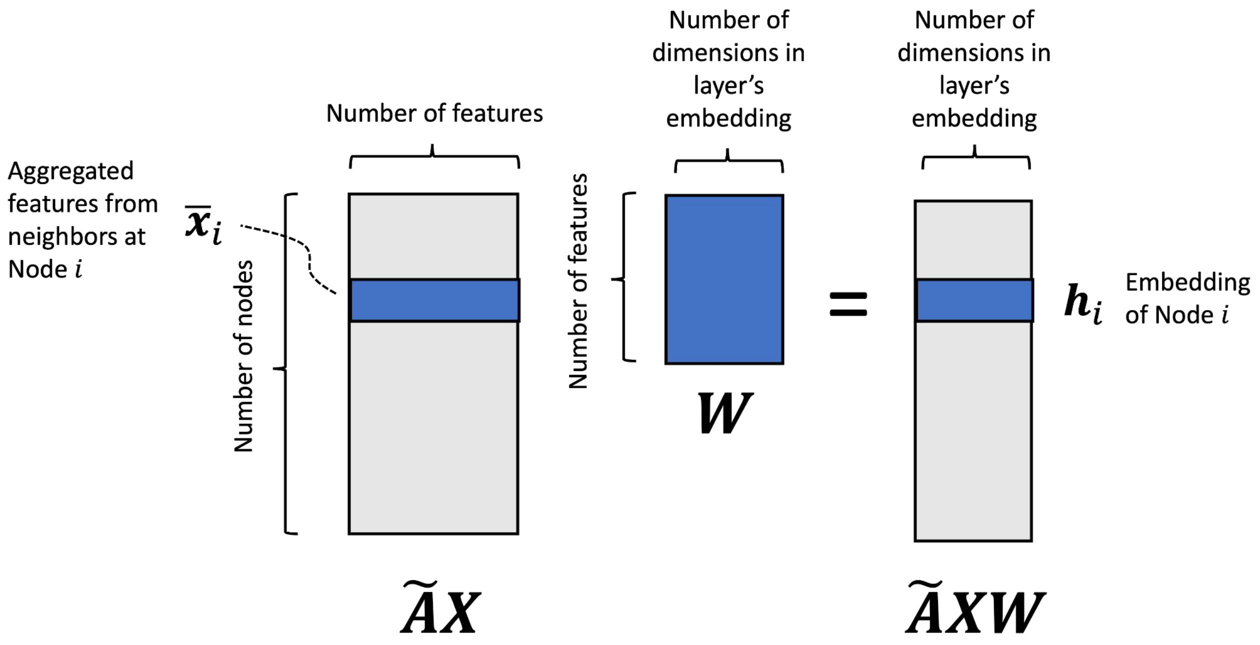
Weighted Sum

$$\bar{\boldsymbol{x}}_i = \sum_{j=1}^{n} \tilde{a_{i,j}} \boldsymbol{x}_j$$

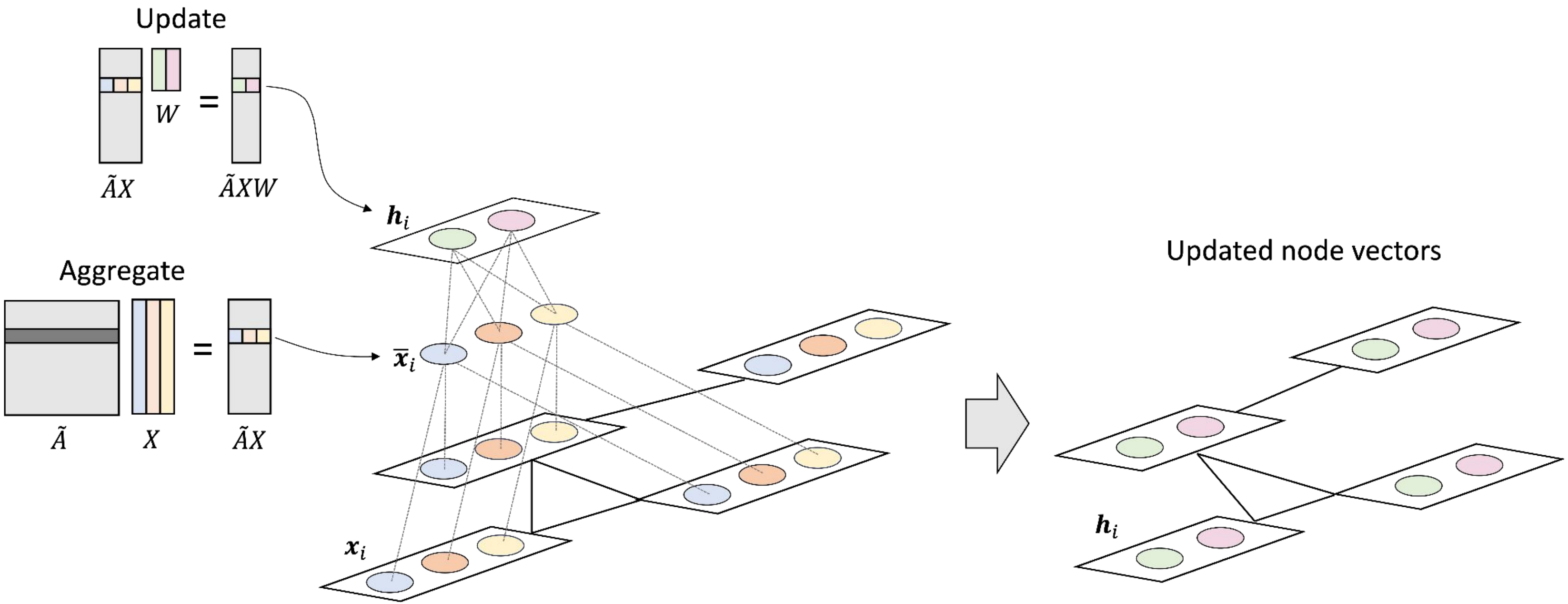
$$= \sum_{j \in Neigh(i)} \tilde{a_{i,j}} \boldsymbol{x}_j$$

$$= \sum_{j \in Neigh(i)} \frac{1}{\sqrt{d_{i,i} d_{j,j}}} \boldsymbol{x}_j$$

Learned weights/parameters

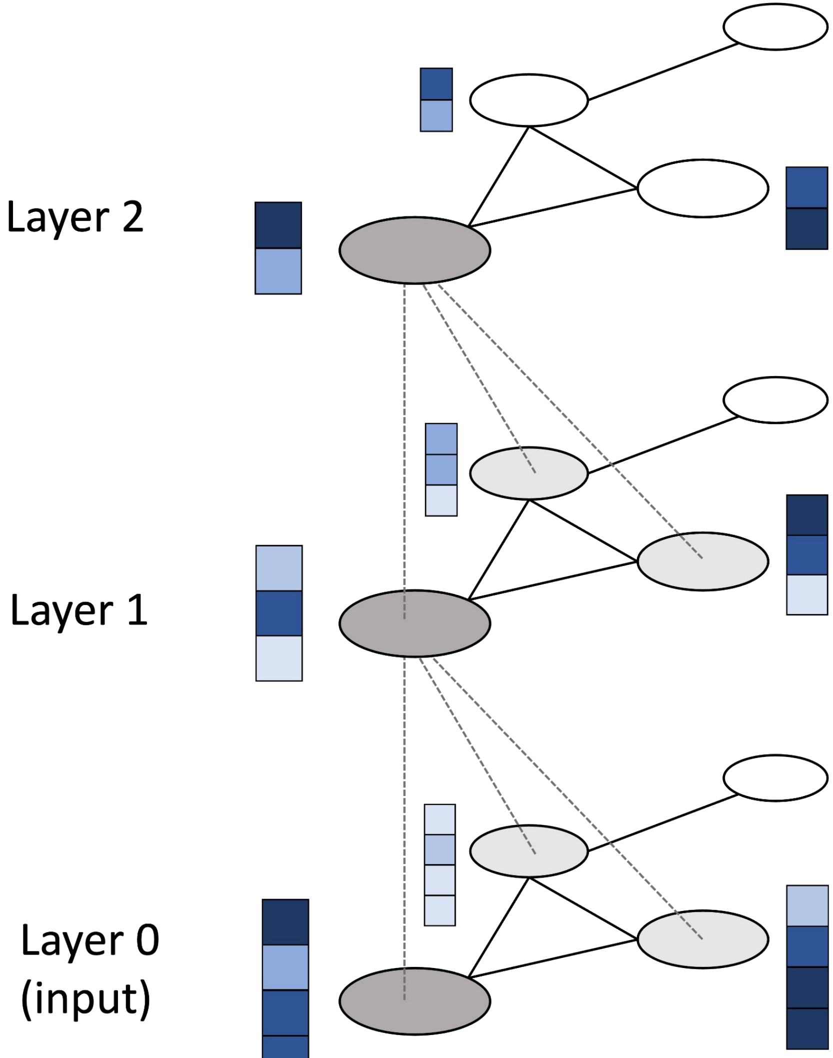


Visualization of the graph convolutional layer



Multi-Layer GCN

$$H_1 := f_{W_1}(X, A)$$
$$H_2 := f_{W_2}(H_1, A)$$
$$H_3 := f_{W_3}(H_2, A)$$



Why to normalize adjacency matrix?

Issue 1: Neighbor count bias

- Nodes with many neighbors → large aggregated vectors
- Nodes with few neighbors → small aggregated vectors
- Leads to embeddings whose magnitudes depend on node degree, not underlying features

Issue 2: Training instability

- Harder to learn weights that detect meaningful patterns independent of neighbor count
- In deeper layers, magnitudes can explode, causing numerical instability

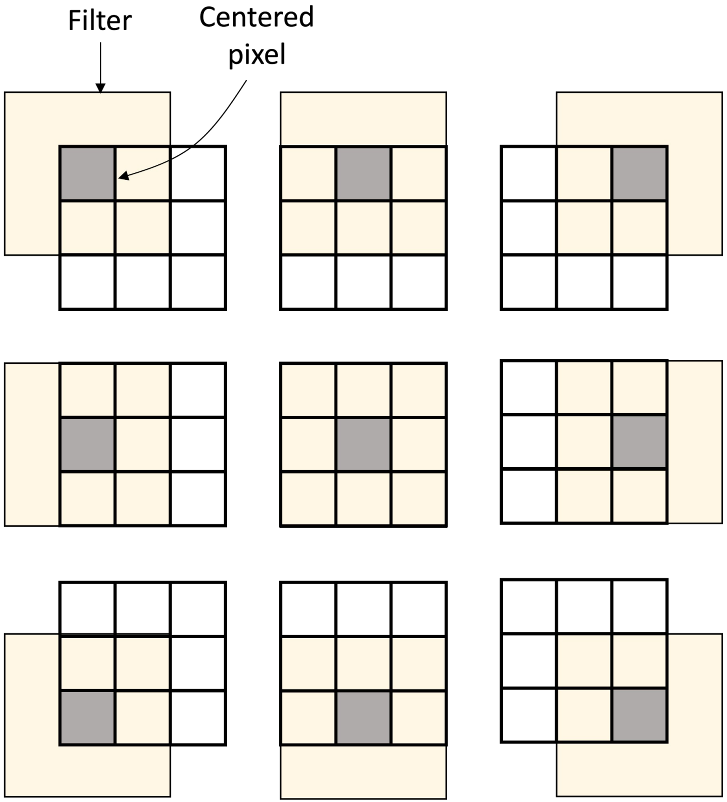
Key takeaway:

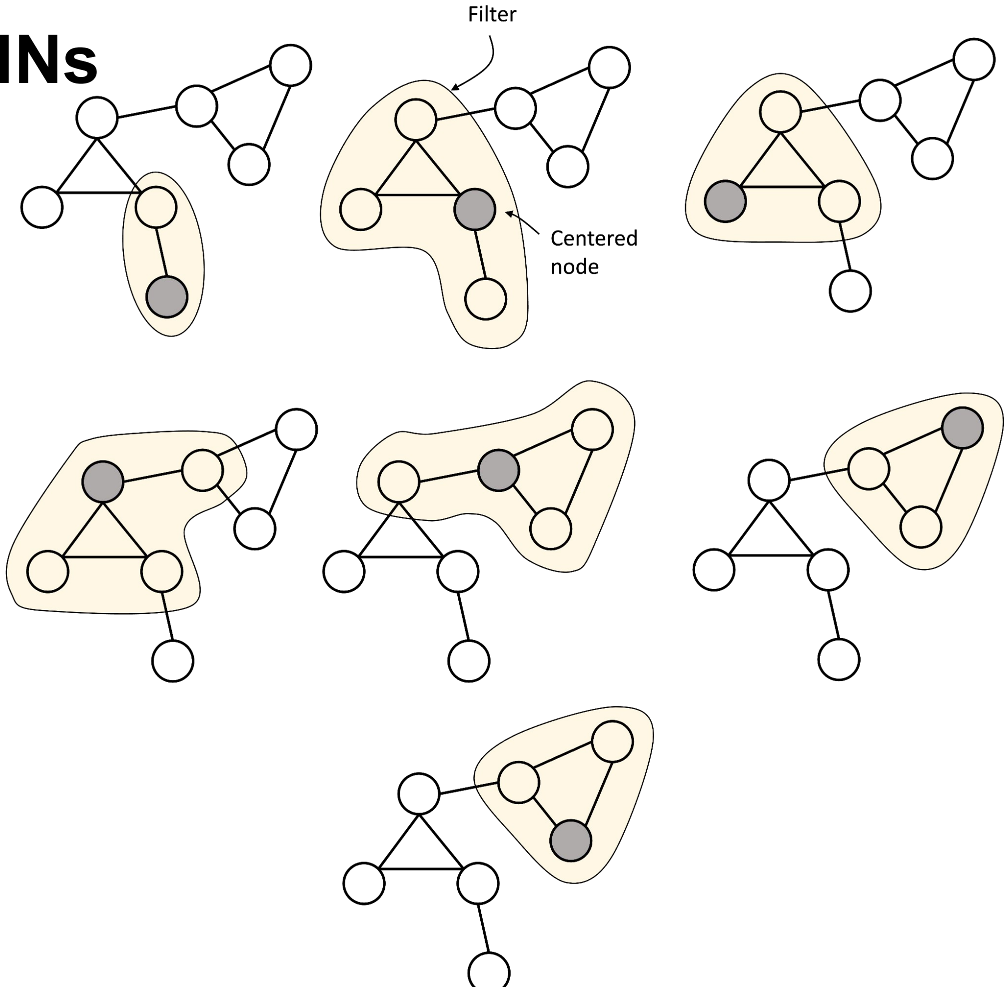
- Need an aggregation method that produces vectors of comparable scale
- Aggregation should be independent of the number of neighbors

Why to normalize adjacency matrix?

- **Example setup:**
  - **Node *i*** has two neighbors:
    - **Neighbor 1:** only connected to *i*
    - **Neighbor 2:** connected to *i* and many others
- **Observation:**
  - Neighbor 2′s message influences **many nodes → stronger impact**
  - Neighbor 1 can only reach the graph **through Node *i* → weaker influence**
- **Problem:**
  - Nodes′ influence is **uneven**, depending on their **degree (number of neighbors)**
- **Solution:**
  - Apply **degree-based normalization** → $\mathbf{D}^{-1/2}\hat{\mathbf{A}}\mathbf{D}^{-1/2}$
  - This **balances** contributions from high- and low-degree nodes
  - Ensures **fair message passing** and **stable training** across the graph

Comparing GCNs and CNNs





Quiz 1

What is the primary data structure GCNs operate on?

A) Images

B) Text sequences

C) Graphs

D) Tabular data

Quiz 2

In a GCN, what does the AGGREGATE step refer to?

A) Combining the node's own features with a neural network

B) Summing or averaging the features of neighboring nodes

C) Applying non-linear activation functions

D) Calculating loss during training

Quiz 3

What is the purpose of the degree matrix in GCNs?

A) To count the number of self-loops in the graph

B) To normalize how neighbor information is weighted based on the number of neighbors

C) To store the feature vectors of nodes

D) To predict node labels

Quiz 4

Why is the adjacency matrix often normalized in GCNs?

A) To simplify matrix multiplication

B) To handle varying node degrees by balancing contributions from nodes with many vs few neighbors

C) To reduce the number of layers in the network

D) To remove edges with low weights

Quiz 5

What determines the "receptive field" (how far information flows) for each node in a GCN?

A) The size of the node features

B) The number of layers in the GCN

C) The activation function used

D) The training data size