Graph Representation Learning

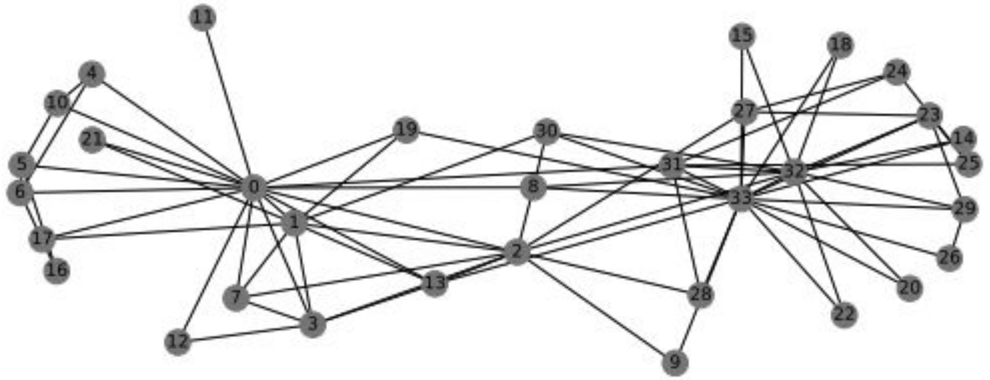
Tawkat, Weirui & Ganesh

Graph Representation Learning - William L. Hamilton
https://www.cs.mcgill.ca/~wlh/grl_book/files/GRL_Book.pdf

Contents

1. Introduction

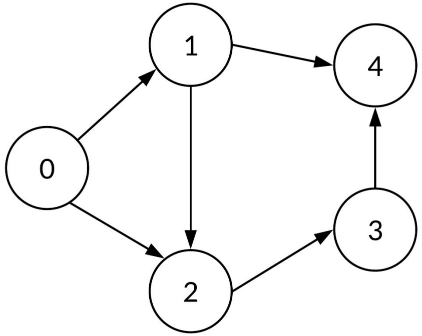
Graph is a collection of **objects** (i.e., nodes) along with a set of **interactions** (i.e., edges) between pairs of these objects.

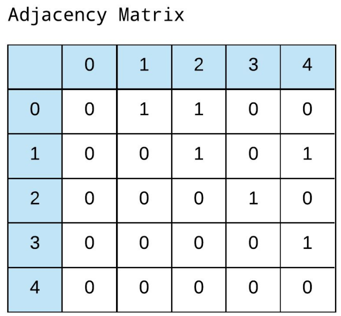


Zachary Karate Club

1. What is a graph?

- Graph, G = (V, E)
- Set of nodes, V
- Set of edges, E
- Edge, (u,v) ∈ E
- Undirected edge, (u,v) ∈ E ↔ (v,u) ∈ E
- Adjacency matrix, A ∈ R^{|V| x |V|}
- A[u,v] = 1 if (u,v) ∈ E and A[u,v] = 0 otherwise
- If the graph contains only undirected edges, what will be the nature of A?
- If edges are weighted, entries in A can be real number



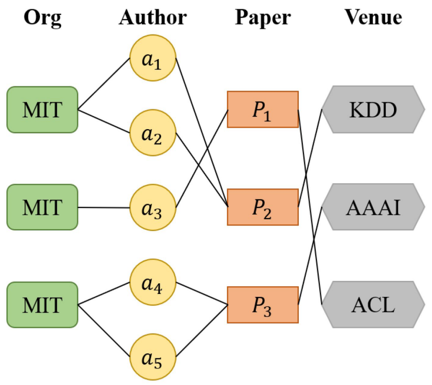

Adjacency Matrix

	0	1	2	3	4
0	0	1	1	0	0
1	0	0	1	0	1
2	0	0	0	1	0
3	0	0	0	0	1
4	0	0	0	0	0

https://guides.codepath.com/compsci/Graphs

1. Multi-relational graphs and feature Information

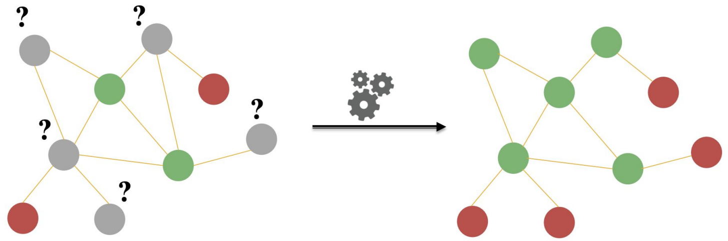
- **Multi-relational graphs**
  - Edges connecting two nodes can have different types (e.g., mutual friends, followers)
  - Edge, (u, τ, v) ∈ E, where τ is edge type
  - Adjacency matrix for a type τ, A_τ
- **Heterogeneous graphs**
  - Nodes also have types (e.g., user-tweet graph)
- **Multiplex graphs**
  - Graph can be decomposed in a set of k layers (e.g., transportation network, node represents a city & each layer represent a different model of transportation)
- **Attribute or feature info.** associated with a graph
  - E.g., profile picture associated with a user in a social network
- Feature matrix, X ∈ R^{|V| x m}



https://www.mdpi.com/2078-2489/12/9/383

1. Machine learning on graphs

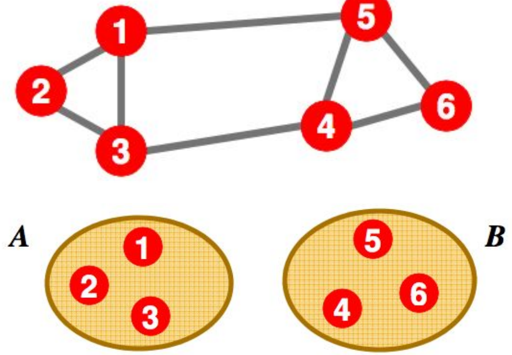
- Node classification
  - Predict label of unseen node, given true labels of a training set of nodes
  - E.g., bot classification
  - Obeys or breaks i.i.d assumption?
  - **Homophily** - similar labels to neighboring nodes in graph
  - **Structural equivalence** - nodes with similar local neighborhood structures have similar labels
  - **Heterophily** - nodes will be preferentially connected to nodes with different labels
- Relation prediction
  - Predict the relation between two nodes, given a training set of relations
  - E.g., recommending content to users



https://snap-stanford.github.io/cs224w-notes

1. Machine learning on graphs

- **Clustering and community detection**
  - Intuition: nodes are more likely to form edges with nodes that belong to the same community
  - E.g., identify communities from collaboration graph
- **Graph classification, regression, and clustering**
  - Predict label for a given graph (typically i.i.d)
  - Graph clustering: learn unsupervised measure of similarity between pairs of graphs



Mining Massive Datasets
Stanford University

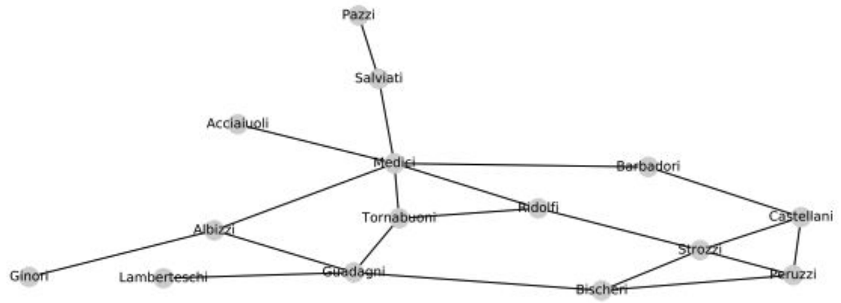
2. Background and Traditional Approaches

Traditional approach

- Extract some statistics or features based on **heuristic function or domain knowledge**
- Use these features as input to a standard ML classifier

Example graph

- Network of 15th century Florentine marriages
  - How Medici family rose to dominate Florentine politics?
  - What features or statistics could a ML model use to predict the Medici's rise?



2. Graph Statistics and Kernel Methods

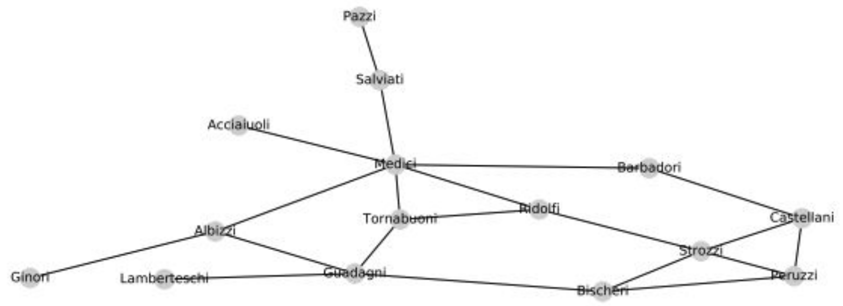
Node degree

$$d_u = \sum_{v \in V} \mathbf{A}[u, v].$$

- Number of edges incident to a node
- Medici family - highest degree in the graph

Node centrality

- **Betweeness centrality** - how often a node lies on the shortest path between two other nodes
- **Closeness centrality** - measures the average shortest path length between a node and all other nodes



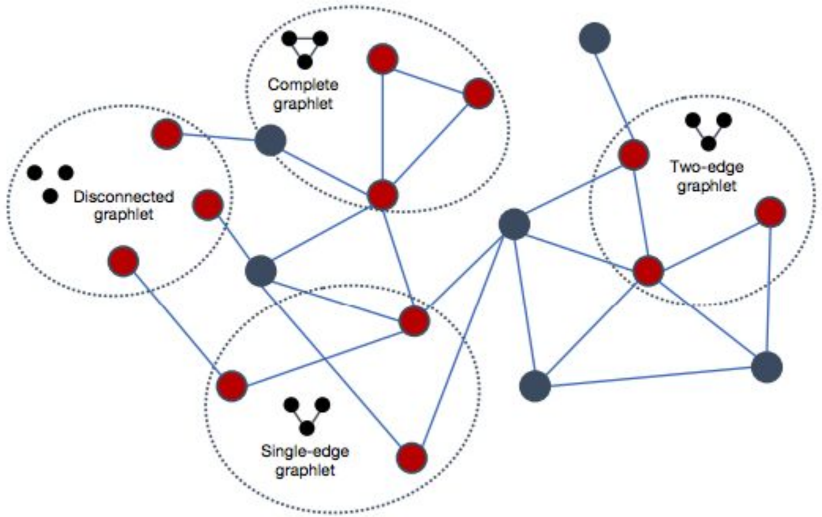
2. Graph-level features and graph kernels

Bag of nodes

- Aggregate node-level statistics: e.g., histogram of node-level statistics
- Downside: only based **on local node-level info**, can miss important global properties

Graphlets method

- Graphlets - different small subgraph structures
- Enumerate all possible graph structures of a particular size and count how many times they occur in the full graph



2. Neighborhood Overlap Detection

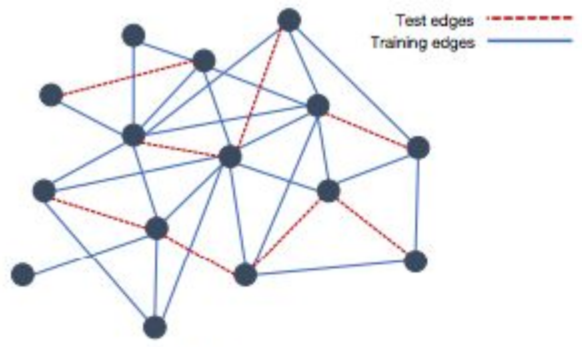
Features that quantify the relationships between nodes

Neighbors overlap:

$$\mathbf{S}[u,v] = |\mathcal{N}(u) \cap \mathcal{N}(v)|,$$

$$P(\mathbf{A}[u,v] = 1) \propto \mathbf{S}[u,v].$$

Local overlap measures: $\mathbf{S}_{\text{Jaccard}}[u,v] = \frac{|\mathcal{N}(u) \cap \mathcal{N}(v)|}{|\mathcal{N}(u) \cup \mathcal{N}(v)|}.$



Training graph

Global overlap measures: Katz Index - Count the number of paths of all lengths between a pair of nodes

$$\mathbf{S}_{\text{Katz}}[u,v] = \sum_{i=1}^{\infty} \beta^i \mathbf{A}^i[u,v], \tag{2.14}$$

where $\beta \in \mathbb{R}^+$ is a user-defined parameter controlling how much weight is given to short versus long paths. A small value of $\beta < 1$ would down-weight the importance of long paths.

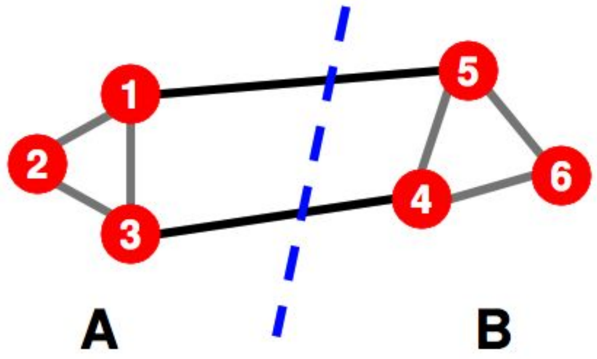
2. Graph Partitioning

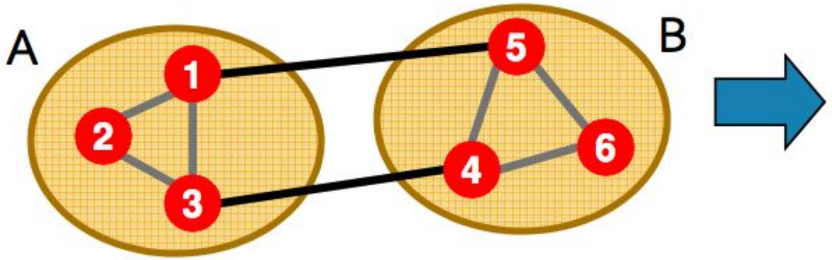
What makes a good partition?

- Maximize the number of within-group connections
- Minimize the number of between-group connections

Cut: Set of edges with only one vertex in a group

$$cut(A,B) = \sum_{i \in A, j \in B} w_{ij}$$

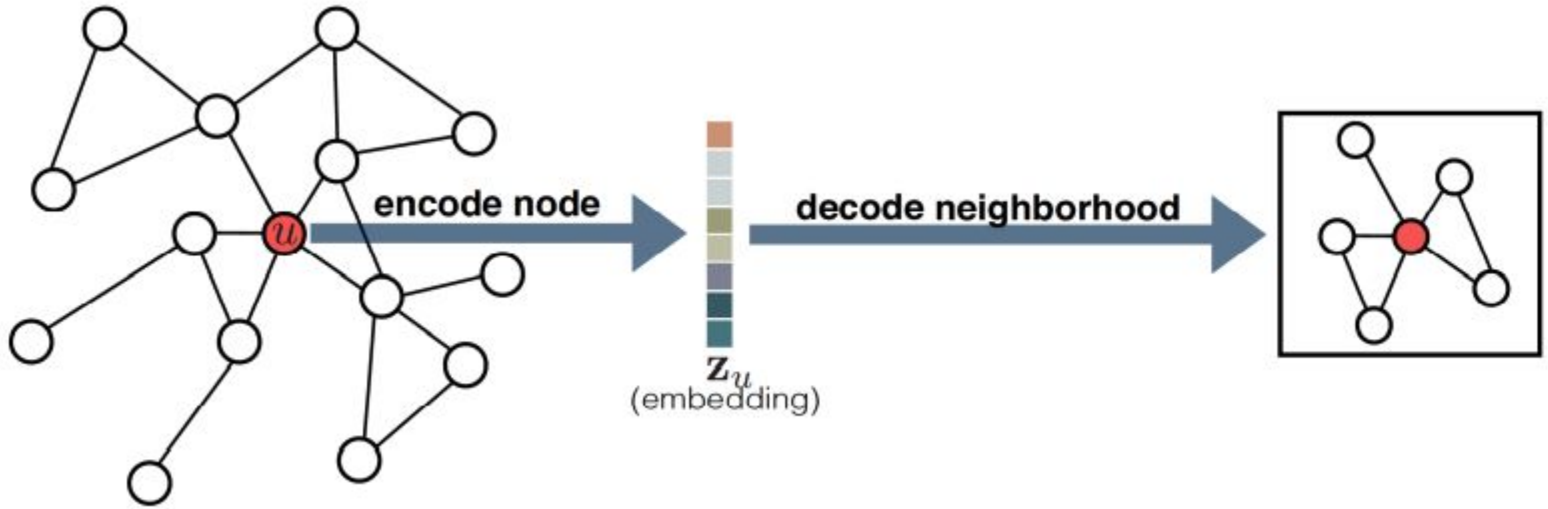




$$cut(A,B) = 2$$

$$\arg\min_{A,B} cut(A,B)$$

3. Encoder-Decoder Architecture



3. Encoder-Decoder Architecture

Encoder: map each vertex to a node embedding

$$\mathrm{ENC}: \mathcal{V} \rightarrow \mathbb{R}^d$$

(Pairwise) Decoder: map each pair of node embeddings to a similarity measure S[u, v]

$$\mathrm{DEC}: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+$$

3. Optimizing Encoder-Decoder Architecture

$$\mathcal{L} = \sum_{(u,v)\in\mathcal{D}} \ell\left(\mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v), \mathbf{S}[u,v]\right)$$

3. Example of Decoder and Loss

$$\mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v) = \|\mathbf{z}_u - \mathbf{z}_v\|_2^2$$

$$\mathcal{L} = \sum_{(u,v)\in\mathcal{D}} \mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v) \cdot \mathbf{S}[u, v]$$

4. Relational-Relational Graph v.s. Single-Relational Graph

Single-Relational Graph

$e = (u, v)$

Multi-Relational Graph

$e=(u, \tau, v)$

4. Multi-Relational Decoder

Now we need to decode differently for each relation

$$\mathrm{DEC} : \mathbb{R}^d \times \mathcal{R} \times \mathbb{R}^d \to \mathbb{R}^+$$

4. Decoder and Loss

$$\mathcal{L} = \sum_{u \in \mathcal{V}} \sum_{v \in \mathcal{V}} \sum_{\tau \in \mathcal{R}} \|\mathrm{DEC}(u, \tau, v) - \mathcal{A}[u, \tau, v]\|^2$$

4. Discussion

Expensive to compute Loss for large graph: $O(|V|^2|R|)$

$$\mathcal{L} = \sum_{u \in \mathcal{V}} \sum_{v \in \mathcal{V}} \sum_{\tau \in \mathcal{R}} \|\mathrm{DEC}(u, \tau, v) - \mathcal{A}[u, \tau, v]\|^2$$

The adjacency tensor is sparse: $|E| << |V|^2|R|$

4. Modified Loss Function

Loss function with Negative sampling

$$\mathcal{L} = \sum_{(u,\tau,v)\in\mathcal{E}} \underbrace{-\log(\sigma(\mathrm{DEC}(\mathbf{z}_u,\tau,\mathbf{z}_v)))}_{\text{Positive datapoints}} \underbrace{- \gamma\mathbb{E}_{v_n\sim P_{n,u}(\mathcal{V})}\left[\log\left(\sigma\left(-\mathrm{DEC}(\mathbf{z}_u,\tau,\mathbf{z}_{v_n})\right)\right)\right]}_{\text{Negative datapoints}}$$

$$\mathrm{DEC}(\mathbf{z}_u,\tau,\mathbf{z}_v) \quad \Uparrow$$

$$\mathrm{DEC}(\mathbf{z}_u,\tau,\mathbf{z}_{v_n}) \quad \Downarrow$$

4. Another example of Modified Loss Function

Max-margin loss

$$\mathcal{L} = \sum_{(u,\tau,v)\in\mathcal{E}} \sum_{v_n\in\mathcal{P}_{n,u}} \max(0, -\overset{\Uparrow}{\mathrm{DEC}}(\mathbf{z}_u, \tau, \mathbf{z}_v) + \underset{\Downarrow}{\mathrm{DEC}(\mathbf{z}_u, \tau, \mathbf{z}_{v_n})} + \Delta)$$

Recap

- In previous chapters, we observed shallow embedding approaches.
- Shallow embedding generates representations of nodes, where we simply optimized a unique embedding vector for each node

Recap

Limitations:

- Do not share any parameters between nodes in the encoder.
  - From a statistical perspective, parameter sharing can improve the efficiency of learning.
  - From the computational perspective, the number of parameters in shallow embedding methods grows as $O(|V|)$, which can be intractable in massive graphs.
- Shallow embedding methods are inherently transductive.
  - Only work on observed nodes.
  - Generating embeddings for new nodes is not possible unless additional optimizations are performed.

5. Intro to Graph Neural Network

Motivations:

- Parameters sharing across the graph.
- Generate representations of nodes depending on the structure of the graph, as well as any feature information.
- Inductive: generalizing to unseen nodes after training.

5. Intro to Graph Neural Network

Challenge with Existing NN:

- Graphs are not of regular shape
  - CNN: Defined over grid-structured inputs (e.g., images).
  - RNN: Defined over grid-structured sequences(e.g., texts).

Why not flatten the graph & feed it to the deep neural network?

$$\mathbf{z}_{\mathcal{G}} = \mathrm{MLP}(\mathbf{A}[1] \oplus \mathbf{A}[2] \oplus \ldots \oplus \mathbf{A}[|\mathcal{V}|]).$$

5. Intro to Graph Neural Network

Challenge with Existing NN:

- Graphs are not of regular shape
    - CNN: Defined over grid-structured inputs (e.g., images).
    - RNN: Defined over grid-structured sequences(e.g., texts).

Why not flatten the graph & feed it to the deep neural network?

$$\mathbf{z}_{\mathcal{G}} = \text{MLP}(\mathbf{A}[1] \oplus \mathbf{A}[2] \oplus \ldots \oplus \mathbf{A}[|\mathcal{V}|]).$$

Depends on the order of the nodes.

5. Intro to Graph Neural Network

Permutation Invariance and Equivariance:

- **Permutation Invariance:** Does NOT depend on the arbitrary ordering of the rows/columns in the adjacency matrix.
- **Permutation Equivariance:** SHOULD BE permuted in an consistent way when we permute the adjacency matrix.

5. Intro to Graph Neural Network

Permutation Invariance and Equivariance:

$$
\begin{aligned}
f(\mathbf{P}\mathbf{A}\mathbf{P}^\top) &= f(\mathbf{A}) && \text{(Permutation Invariance)} \\
f(\mathbf{P}\mathbf{A}\mathbf{P}^\top) &= \mathbf{P}f(\mathbf{A}) && \text{(Permutation Equivariance)}
\end{aligned}
$$

Where, P is a permutation matrix.

Permutation Matrix: A square binary matrix that has exactly one entry of 1 in each row and each column and 0s elsewhere.

Our function for GNN should satisfy one of the above properties.

5.1 Neural Message Passing

Intuition

- At each iteration, every node should aggregate information from its local neighborhood.
- As the iterations progress, each node embedding should contain more and more information from further reaches of the graph.
    - after the first iteration (k = 1), every node embedding contains information from its 1-hop neighborhood (immediate neighbors).
    - after the second iteration (k = 2), every node embedding contains information from its 2-hop neighborhood.
    - after k iterations, every node embedding contains information about its k-hop neighborhood.

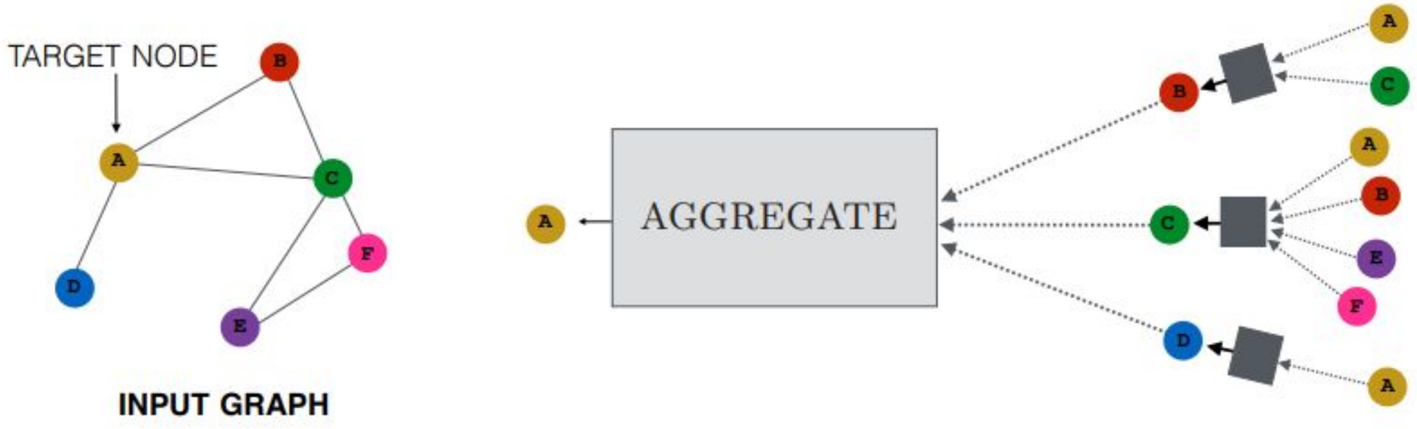
5.1 Neural Message Passing

Common notations:

- Graph, $G = (V, E)$
- Node features set, $\mathbf{X} \in R^{d \times |V|}$
- Node embedding, z_u, $\forall u \in V$

5.1 Neural Message Passing

Overview:



5.1 Neural Message Passing

Overview:

$$\begin{aligned}\mathbf{h}_u^{(k+1)} &= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})\right) \\ &= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right),\end{aligned}$$

$$\mathbf{z}_u = \mathbf{h}_u^{(K)}, \forall u \in \mathcal{V}.$$

5.1 Neural Message Passing

What information node embeddings encode?:

- **Structural Information:** After k iterations, the embedding $h^{(k)}(u)$ of node u might encode information about the degrees of all the nodes in u’s k-hop neighborhood.
- **Feature Aggregation:** After k iterations, the embeddings for each node also encode information about all the features in their k-hop neighborhood.

5.1 Neural Message Passing

Basic GNN:

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}_{\text{self}}^{(k)}\mathbf{h}_u^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)}\sum_{v\in\mathcal{N}(u)}\mathbf{h}_v^{(k-1)} + \mathbf{b}^{(k)}\right),$$

where $\mathbf{W}_{\text{self}}^{(k)}, \mathbf{W}_{\text{neigh}}^{(k)} \in \mathbb{R}^{d^{(k)}\times d^{(k-1)}}$ are trainable parameter matrices and σ denotes an elementwise non-linearity (e.g., a tanh or ReLU). The bias term $\mathbf{b}^{(k)} \in \mathbb{R}^{d^{(k)}}$

5.1 Neural Message Passing

Basic GNN:

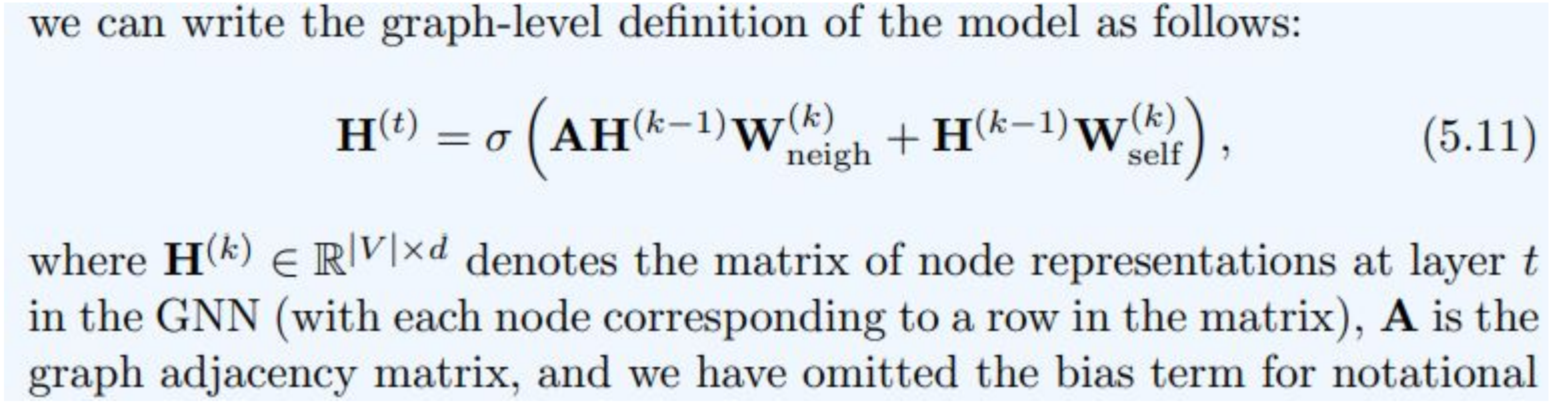
$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v,$$

$$\text{UPDATE}(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}) = \sigma\left(\mathbf{W}_{\text{self}}\mathbf{h}_u + \mathbf{W}_{\text{neigh}}\mathbf{m}_{\mathcal{N}(u)}\right),$$

$$\mathbf{m}_{\mathcal{N}(u)} = \text{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})$$

5.1 Neural Message Passing

Basic GNN (Graph level Equation):



we can write the graph-level definition of the model as follows:

$$\mathbf{H}^{(t)} = \sigma\left(\mathbf{A}\mathbf{H}^{(k-1)}\mathbf{W}_{\text{neigh}}^{(k)} + \mathbf{H}^{(k-1)}\mathbf{W}_{\text{self}}^{(k)}\right), \tag{5.11}$$

where $\mathbf{H}^{(k)} \in \mathbb{R}^{|V|\times d}$ denotes the matrix of node representations at layer t in the GNN (with each node corresponding to a row in the matrix), $\mathbf{A}$ is the graph adjacency matrix, and we have omitted the bias term for notational

5.1 Neural Message Passing

Message Passing with Self-Loops:

$$\mathbf{h}_u^{(k)} = \text{AGGREGATE}(\{\mathbf{h}_v^{(k-1)}, \forall v \in \mathcal{N}(u) \cup \{u\}\}),$$

where now the aggregation is taken over the set $\mathcal{N}(u) \cup \{u\}$,

$$\mathbf{H}^{(t)} = \sigma\left((\mathbf{A} + \mathbf{I})\mathbf{H}^{(t-1)}\mathbf{W}^{(t)}\right).$$

In the following chapters we will refer to this as the *self-loop* GNN