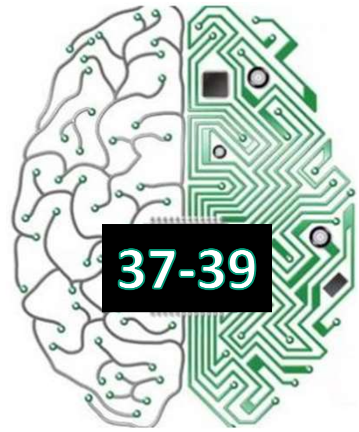
Open Elective Course [OE]
Course Code: CSO507
Winter 2023-24

Lecture#

Deep Learning

Unit-8: Few more GAN variants
Unit-9: Graph Neural Networks

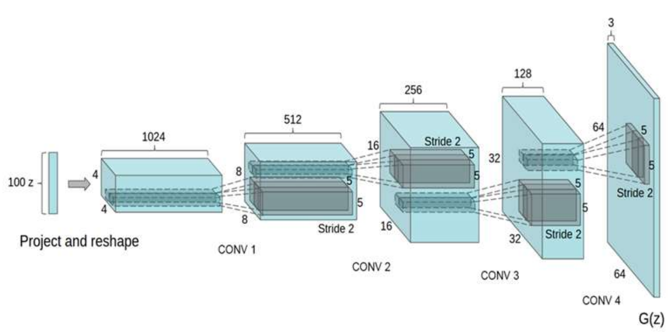


Course Instructor:
Dr. Monidipa Das
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology (Indian School of Mines) Dhanbad, Jharkhand 826004, India

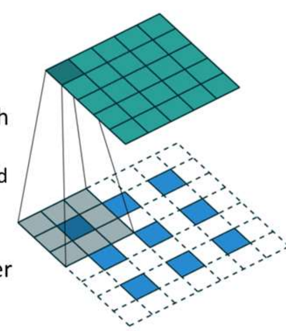
Deep Convolutional GANs (DCGANs)

2

Generator Architecture



Key ideas:
- Replace FC hidden layers with Convolutions
  - **Generator:** Fractional-Strided convolutions
- Use Batch Normalization after each layer
- **Inside Generator**
  - Use ReLU for hidden layers
  - Use Tanh for the output layer



Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).

LSGAN

3

- proposes to use the least-squares loss function for the discriminator.

Vanilla GAN:

$$\min_G \max_D V_{\text{GAN}}(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

LSGAN:

$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x}) - b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})) - a)^2\right]$$

$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})) - c)^2\right],$$

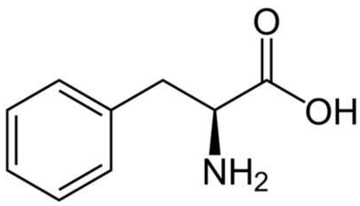
4

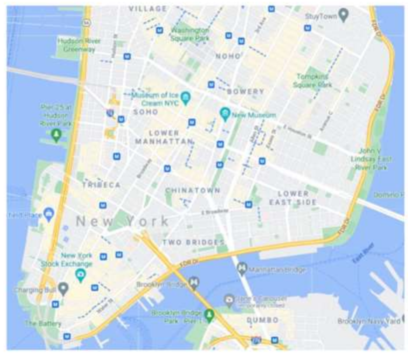
Unit-9: Graph Neural Networks

Acknowledgement: Prof. E. Knag et al., Prof. J. Leskovec

Graphs in the World

5





phenylalanine

Map of Manhattan

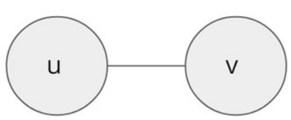
Social Network

Graph Definitions

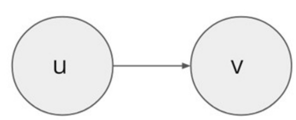
6

G = (V, E)

- V is a set of nodes
- E is a set of tuples of form (u, v), where there is an edge between u and v
- G is a graph



Undirected edge



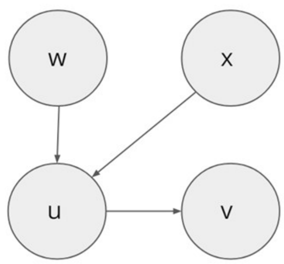
Directed edge

Graph encoding as a matrix

Adjacency Matrix: $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$

- In this example, binary matrix encoding of a unweighted graph
- Rows/columns number the nodes, matrix elements encode edges

V = {u, v, w, x}; E = {(w, u), (x, u), (u, v)}



A = (to: columns; from: rows)

	u	v	w	x
u	0	**1**	0	0
v	0	0	0	0
w	**1**	0	0	0
x	**1**	0	0	0

Do the matrices encode the same graph?

0	**1**	0	0
0	0	0	0
1	0	0	0
1	0	0	0

0	0	0	0
0	0	**1**	0
1	0	0	0
0	0	**1**	0

Do the matrices encode the same graph?

9

0	1	0	0
0	0	0	0
1	0	0	0
1	0	0	0

0	0	0	0
0	0	1	0
1	0	0	0
0	0	1	0

u	v	w	x	
0	1	0	0	u
0	0	0	0	v
1	0	0	0	w
1	0	0	0	x

v	w	u	x	
0	0	0	0	v
0	0	1	0	w
1	0	0	0	u
0	0	1	0	x

Do the matrices encode the same graph?

10

u	v	w	x	
0	1	0	0	u
0	0	0	0	v
1	0	0	0	w
1	0	0	0	x

v	w	u	x	
0	0	0	0	v
0	0	1	0	w
1	0	0	0	u
0	0	1	0	x

Considerations for GNN

11

- Nodes are not i.i.d
- A NN modeling a graph should be permutation invariant and equivariant ○ Adjacency matrix orders nodes arbitrarily

For permutation matrix **P**, function *f* that takes in an adjacent matrix **A**:

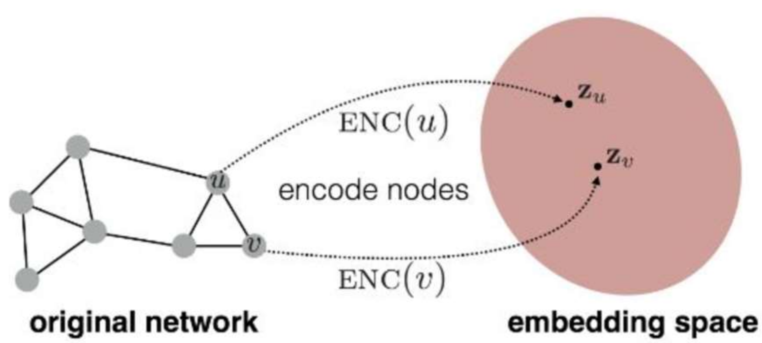
Permutation Invariance Property: $f(\mathbf{PAP^T}) = f(\mathbf{A})$

Permutation Equivariance Property: $f(\mathbf{PAP^T}) = \mathbf{P}f(\mathbf{A})$

Considerations for GNN

12

3) Find an encoding that <u>preserves the graph structure</u>



Neural Message Passing

13

Graph

$$G = (V, E)$$

Node Features

$$\boldsymbol{X} \in \mathbb{R}^{d \times |V|}$$

Node Embeddings

$$\boldsymbol{z_u}, \forall u \in V$$

Hidden embedding:

$$\boldsymbol{h} = \{\overrightarrow{h_1}, \overrightarrow{h_2}, \ldots, \overrightarrow{h_N}\}$$

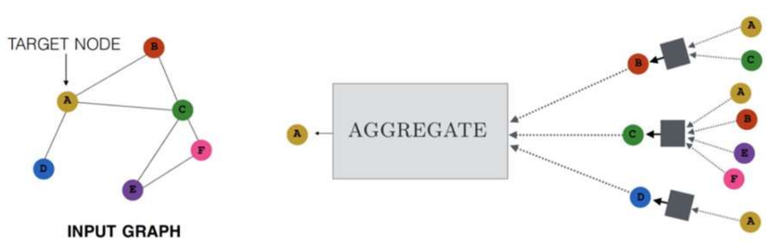
Node vs Edge embeddings:

$$h_u^{(k)}, u \in V$$

$$h_{(u,v)}^{(k)}, (u, v) \in E$$

Message Passing Framework

14



$$h_u^{(k+1)} = \text{UPDATE}^{(k)}\left(h_u^{(k)}, \text{AGGREGATE}^{(k)}\left(\{h_v^{(k)}, \forall v \in \mathcal{N}(u)\}\right)\right)$$

$$h_u^{(k+1)} = \text{UPDATE}^{(k)}\left(h_u^{(k)}, m_{\mathcal{N}(u)}^{(k)}\right)$$

Message Passing Framework

15

AT EACH ITERATION k OF THE GNN:

- $\textit{AGGREGATE}$ all embeddings from u's neighbors to generate a message $m_{\mathcal{N}(u)}^{(k)}$ based on this aggregated neighborhood information
- $\textit{UPDATE}$ the embedding $\boldsymbol{h}_{\boldsymbol{u}}^{(k+1)}$ of node u by combining information from the previous embedding $\boldsymbol{h}_{\boldsymbol{u}}^{(k)}$ and with the message $m_{\mathcal{N}(u)}^{(k)}$

Message Passing Framework

16

AFTER RUNNING K ITERATIONS:

- Use the output of the final layer to define the embeddings for each node:

$$z_u = h_u^{(K)}, \forall u \in V$$

The Basic GNN

17

$$h_u^{(k)} = \sigma\left(W_{\text{self}}^{(k)} h_u^{(k-1)} + W_{\text{neigh}}^{(k)} \sum_{v \in N_u} h_v^{(k-1)} + b^{(k)}\right)$$

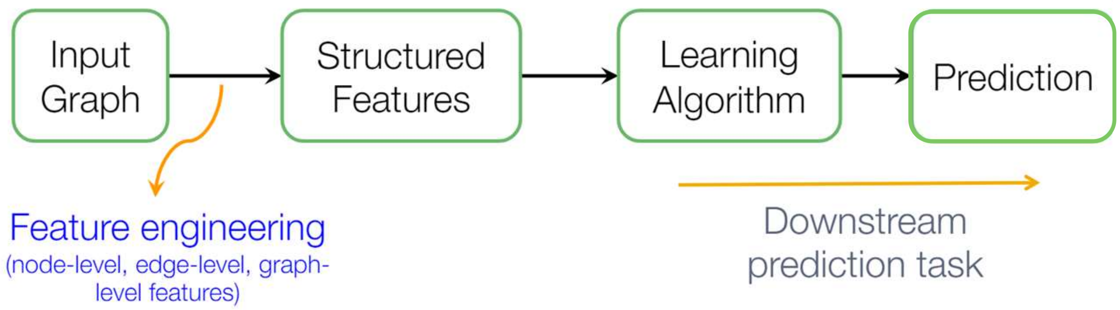
- $h_u^{(k-1)} \in \mathbb{R}^{d^{(k-1)}}$: Node embeddings
- $W_{\text{self}}^{(k)}, W_{\text{neigh}}^{(k)} \in \mathbb{R}^{d^{(k)} \times d^{(k-1)}}$: Learnable parameters
- $b^{(k)} \in \mathbb{R}^{d^{(k)}}$: Bias term
- σ: Elementwise non-linearity (e.g., a tanh or ReLU)

Summary

18

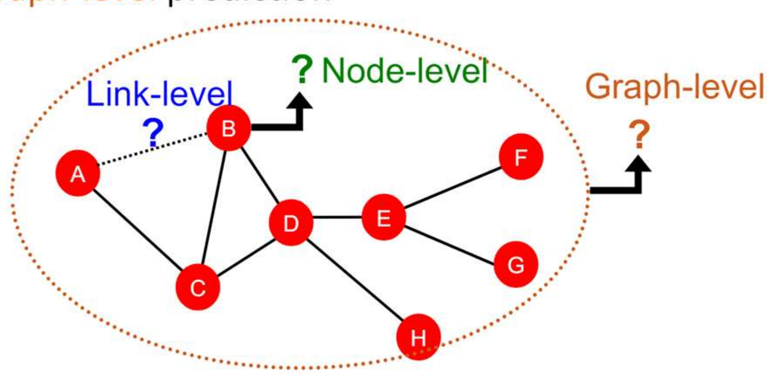
1. Sum the messages incoming from the neighbors
2. Combine the neighborhood information with the node's previous embedding using a linear combination
3. Apply an elementwise non-linearity

Traditional Machine Learning on Graphs



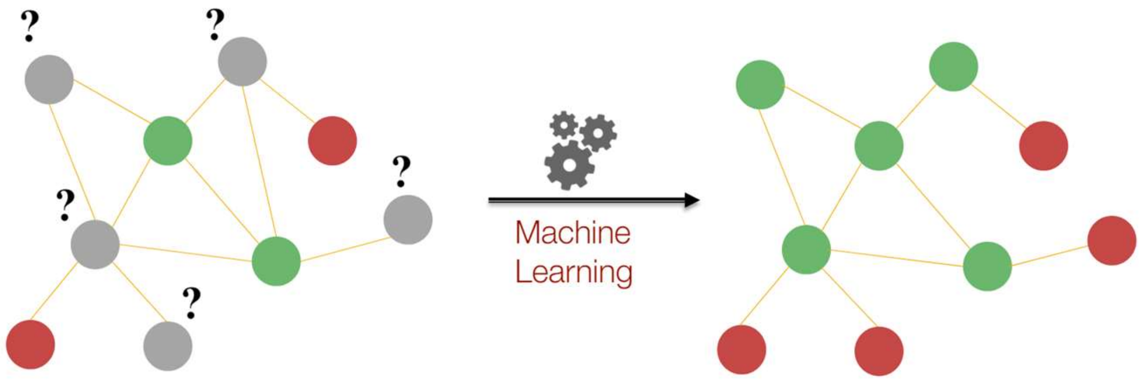
Learning Tasks

- Node-level prediction
- Link-level prediction
- Graph-level prediction



Node-Level Prediction Tasks

21

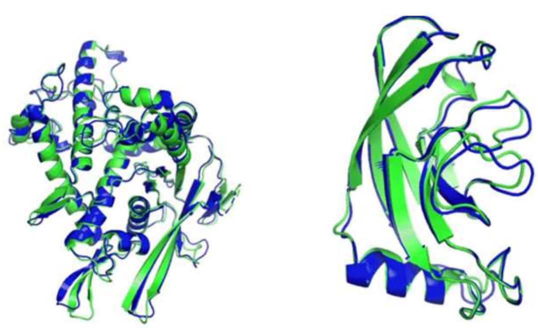


Node classification

Example

22

- Predict a protein's 3D structure based solely on its amino acid sequence

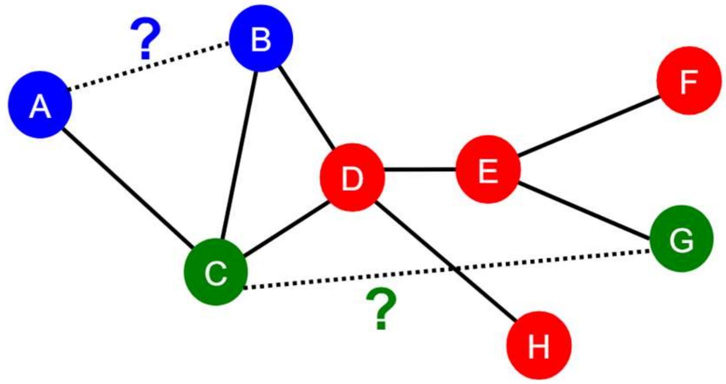


T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)

T1049 / 6y4f
93.3 GDT
(adhesin tip)

Link-Level Prediction Tasks

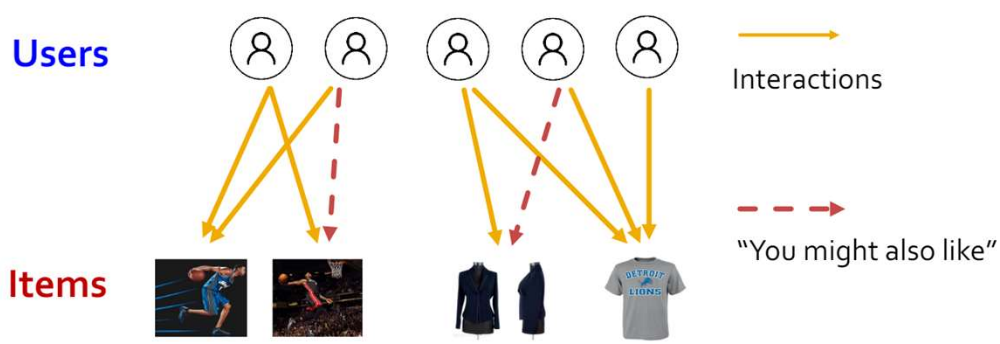
23



Example-1

24

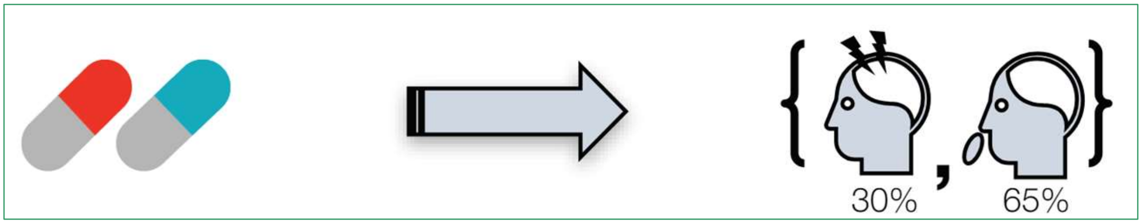
- Recommender Systems



Example-2

25

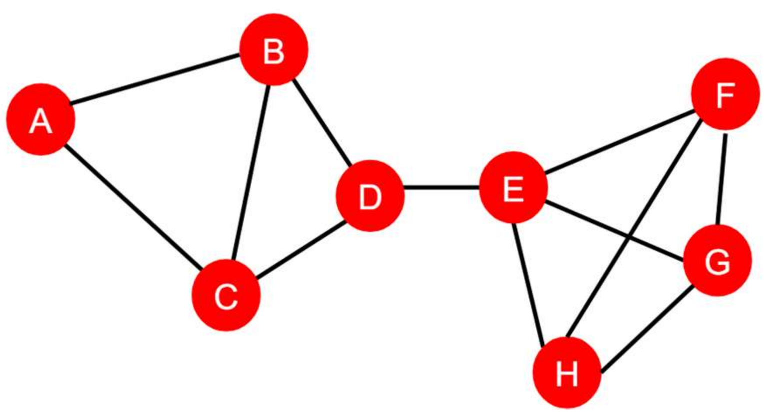
- Given a pair of drugs predict adverse side effects



Graph-Level Tasks

26

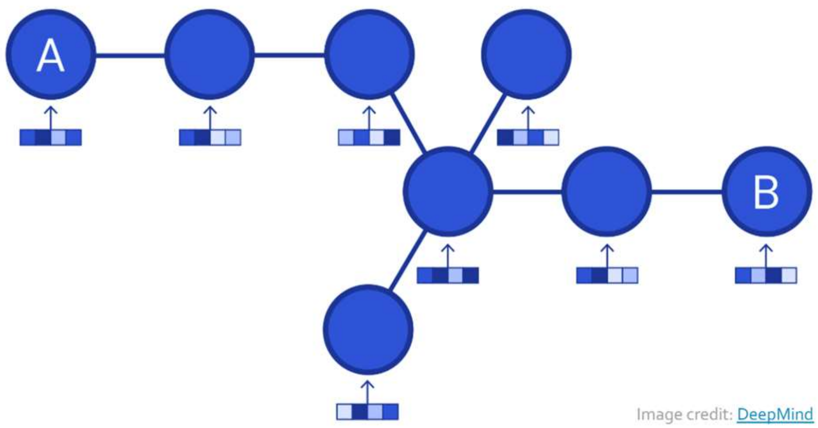
- Graph/Sub-graph Classification



Example-1

27

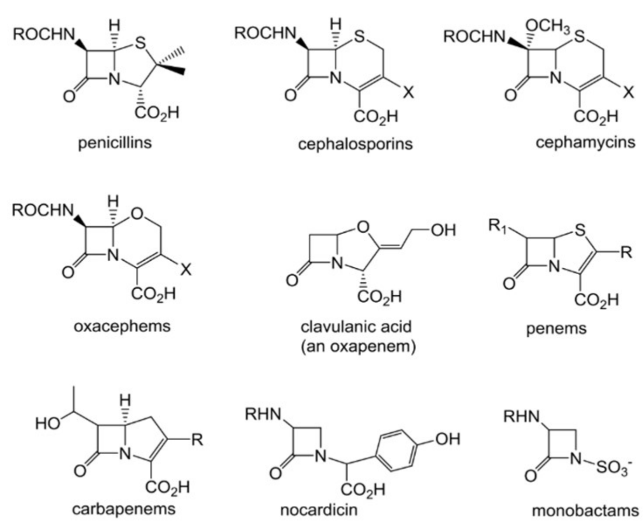
- Predicting Arrival Time



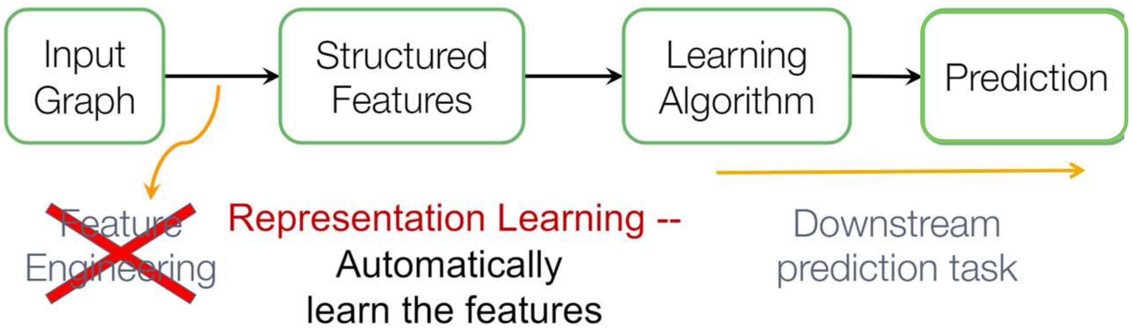
Example-2

28

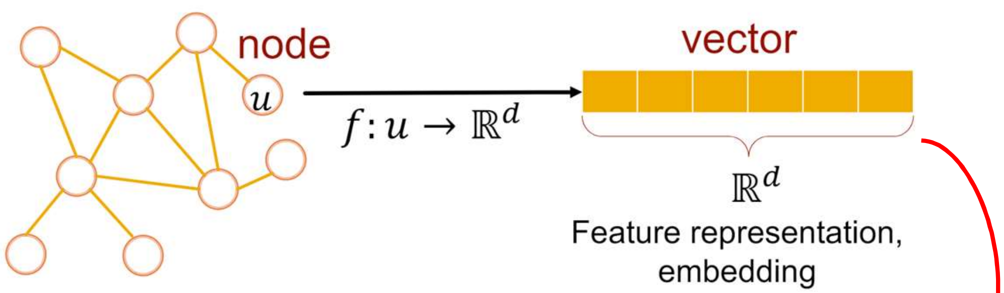
- Drug Discovery



Graph Representation Learning



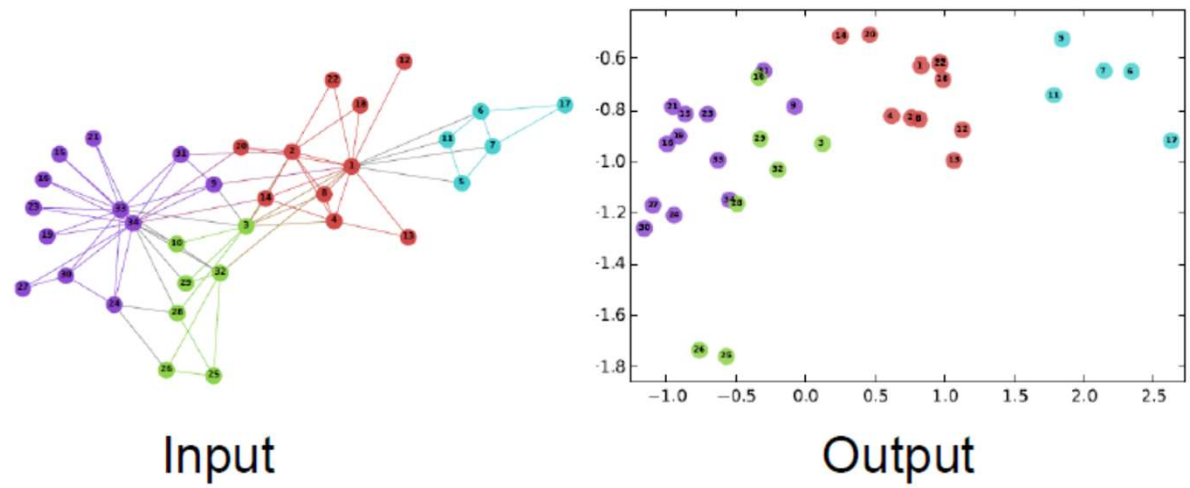
Graph Representation Learning



Tasks
- Node classification
- Link prediction
- Graph classification
- Anomalous node detection
- Clustering
- ….

Example: Node Embedding

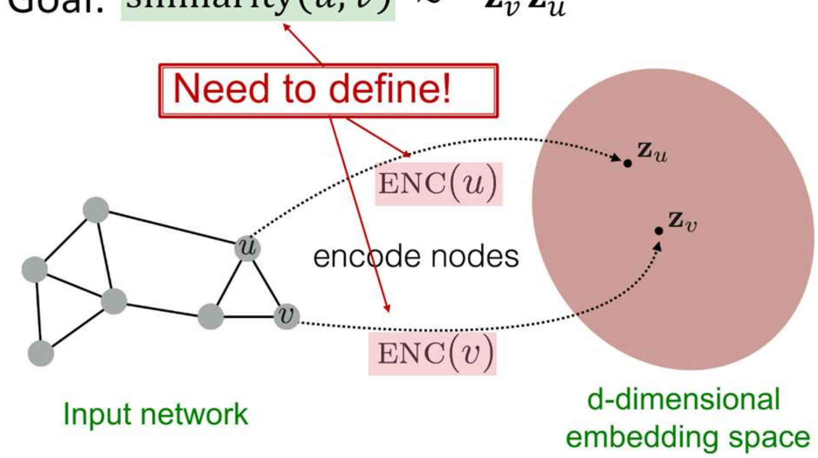
31



Node Embedding

32

Goal: $\text{similarity}(u, v) \approx \mathbf{z}_v^{\mathrm{T}} \mathbf{z}_u$



Learning Node Embedding

1. **Encoder** maps from nodes to embeddings
2. **Define a node similarity function** (i.e., a measure of similarity in the original network)
3. **Decoder DEC** maps from embeddings to the similarity score
4. **Optimize the parameters of the encoder so that:**

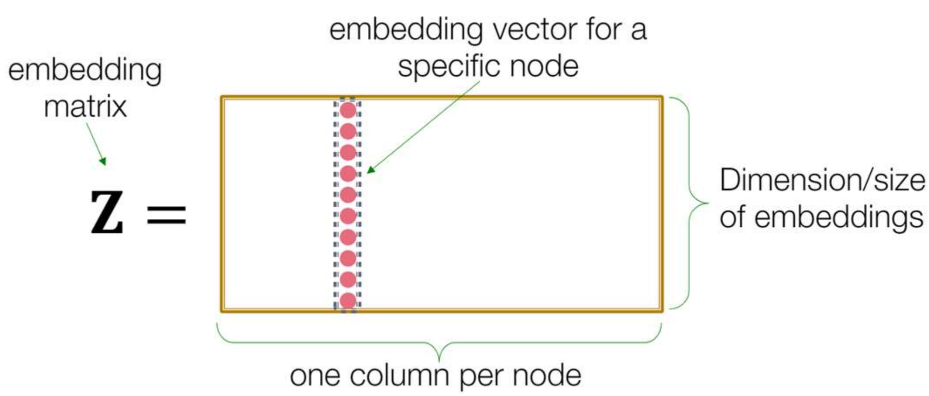
$\text{DEC}(\mathbf{z}_v^{\mathrm{T}}\mathbf{z}_u)$

$$\underbrace{\text{similarity}(u, v)}_{\text{in the original network}} \approx \underbrace{\mathbf{z}_v^{\mathrm{T}}\mathbf{z}_u}_{\text{Similarity of the embedding}}$$

Simplest Embedding

Simplest encoding approach: **Encoder is just an embedding-lookup**

$\mathbf{Z} =$



Limitation

35

- Limitations of shallow embedding methods:
  - Large number of parameters are needed
    - No sharing of parameters between nodes
    - Every node has its own unique embedding
  - Inherently "transductive"
    - Cannot generate embeddings for nodes that are not seen during training
  - Do not incorporate node features
    - Nodes in many graphs have features that we can and should leverage

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Deep Graph Encoders

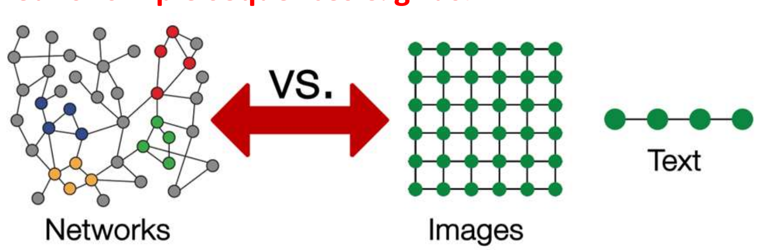
36

- Based on deep learning for graphs (using **Graph Neural Networks**)

$\mathrm{ENC}(v) =$ **multiple layers of non-linear transformations based on graph structure**

Why GNNs?

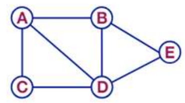
The CNNs and RNNs are designed for simple sequences & grids. Difficult to handle graphs!



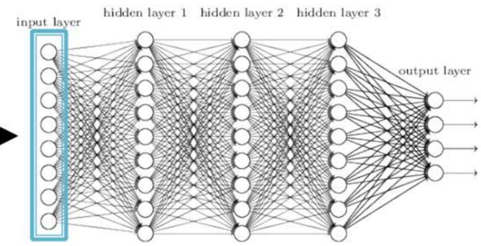
Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Problems with Approaches Learnt So Far

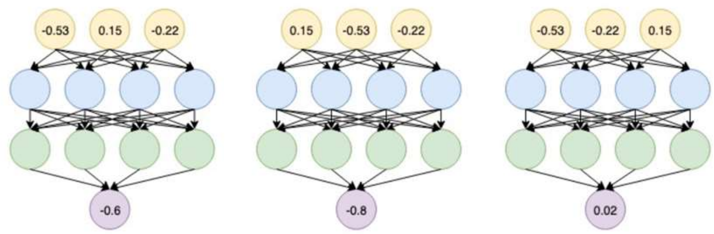
37



	A	B	C	D	E	Feat	
A	0	1	1	1	0	1	0
B	1	0	0	1	1	0	0
C	1	0	0	1	0	0	1
D	1	1	1	0	1	1	1
E	0	1	0	1	0	1	0



?

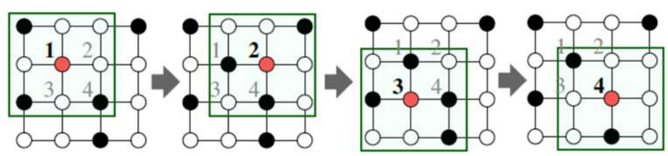


Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

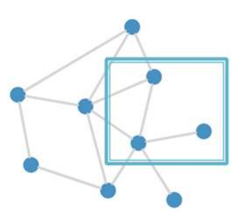
Problems with Approaches Learnt So Far

38

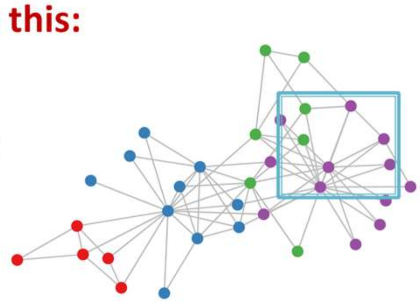
CNN on an image:



But our graphs look like this:



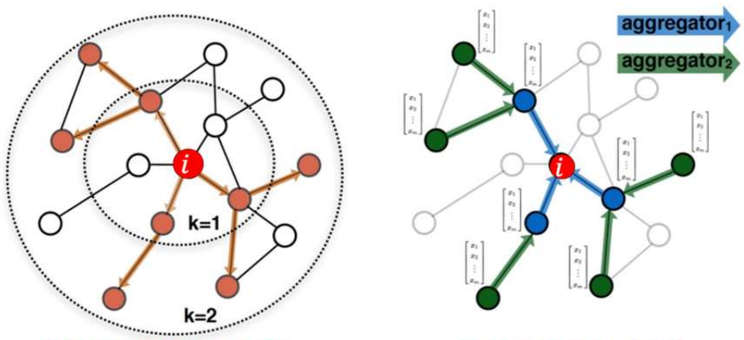
or this:



Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Graph Convolutional Networks

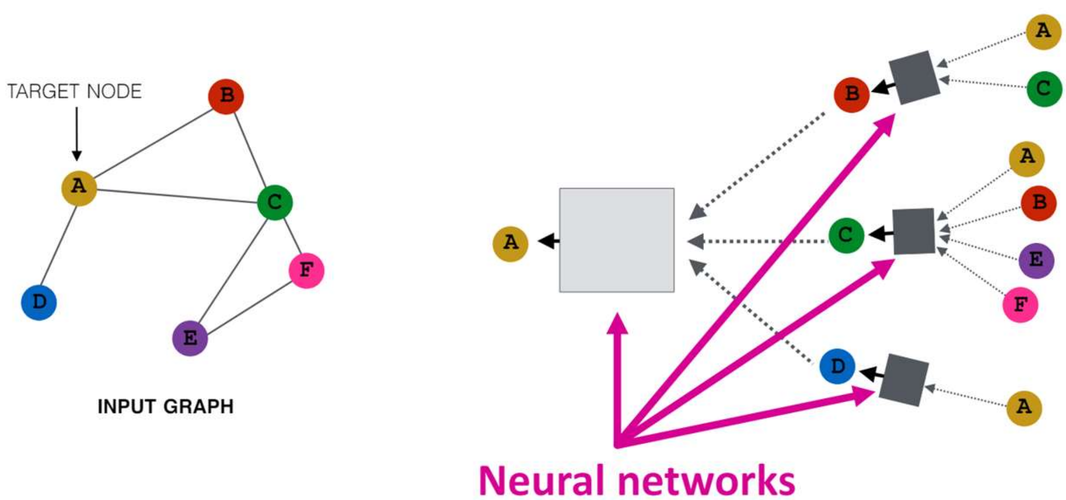
- **Idea:** Node's neighborhood defines a computation graph



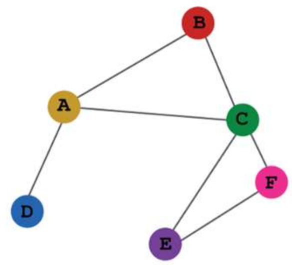
Determine node computation graph

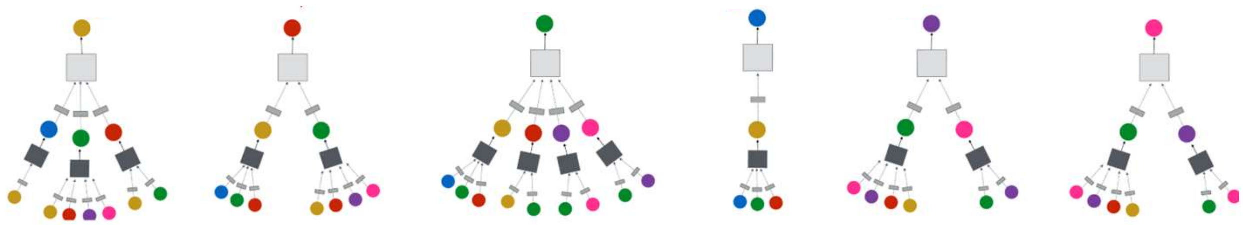
Propagate and transform information

Aggregate Neighbors



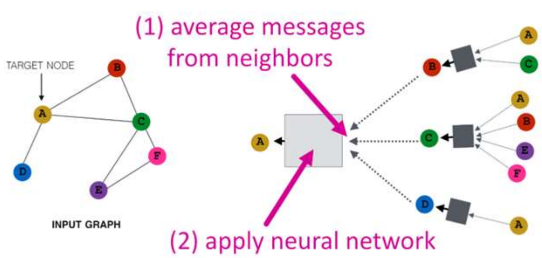
Aggregate Neighbors





Node Aggregation: Basic Approach

Average neighbor messages and apply a neural network



Initial 0-th layer embeddings are equal to node features

$$\boxed{h_v^0 = x_v}$$

embedding of v at layer k

$$h_v^{(k+1)} = \boxed{\sigma}\left(W_k \boxed{\sum_{u \in N(v)} \frac{h_u^{(k)}}{|N(v)|}} + B_k \boxed{h_v^{(k)}}\right), \forall k \in \{0, \dots, \boxed{K} - 1\}$$

$$\boxed{z_v} = h_v^{(K)}$$

Embedding after K layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

Average of neighbor's previous layer embeddings

Total number of layers

Notice summation is a permutation invariant pooling/aggregation.

Graph Convolutional Networks (GCN)

GraphSAGE

43

$$\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)} \cdot \text{CONCAT}\left(\mathbf{h}_v^{(l-1)}, \text{AGG}\left(\left\{\mathbf{h}_u^{(l-1)}, \forall u \in N(v)\right\}\right)\right)\right)$$

- **How to write this as Message + Aggregation?**
  - **Message** is computed within the $\text{AGG}(\cdot)$
  - **Two-stage aggregation**
    - **Stage 1:** Aggregate from node neighbors
      $$\mathbf{h}_{N(v)}^{(l)} \leftarrow \text{AGG}\left(\left\{\mathbf{h}_u^{(l-1)}, \forall u \in N(v)\right\}\right)$$
    - **Stage 2:** Further aggregate over the node itself
      $$\mathbf{h}_v^{(l)} \leftarrow \sigma\left(\mathbf{W}^{(l)} \cdot \text{CONCAT}(\mathbf{h}_v^{(l-1)}, \mathbf{h}_{N(v)}^{(l)})\right)$$

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

GraphSAGE

44

- **Mean:** Take a weighted average of neighbors
  $$\text{AGG} = \underbrace{\sum_{u \in N(v)}}_{\text{Aggregation}} \frac{\mathbf{h}_u^{(l-1)}}{\underbrace{|N(v)|}_{\text{Message computation}}}$$
- **Pool:** Transform neighbor vectors and apply symmetric vector function $\text{Mean}(\cdot)$ or $\text{Max}(\cdot)$
  $$\text{AGG} = \underbrace{\text{Mean}}_{\text{Aggregation}}(\{\underbrace{\text{MLP}}_{\text{Message computation}}(\mathbf{h}_u^{(l-1)}), \forall u \in N(v)\})$$
- **LSTM:** Apply LSTM to reshuffled of neighbors
  $$\text{AGG} = \underbrace{\text{LSTM}}_{\text{Aggregation}}([\mathbf{h}_u^{(l-1)}, \forall u \in \pi(N(v))])$$

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

GraphSAGE

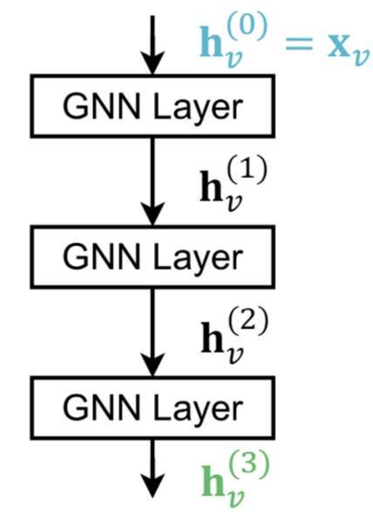
45

- ℓ_2 **Normalization:**
  - **Optional:** Apply ℓ_2 normalization to $\mathbf{h}_v^{(l)}$ at every layer
  - $\mathbf{h}_v^{(l)} \leftarrow \frac{\mathbf{h}_v^{(l)}}{\left\|\mathbf{h}_v^{(l)}\right\|_2} \; \forall v \in V$ where $\|u\|_2 = \sqrt{\sum_i u_i^2}$ (ℓ_2-norm)
  - Without ℓ_2 normalization, the embedding vectors have different scales (ℓ_2-norm) for vectors
  - In some cases (not always), normalization of embedding results in performance improvement
  - After ℓ_2 normalization, all vectors will have the same ℓ_2-norm

Constructing Graph Neural Network

46

- Stacking GNN Layers

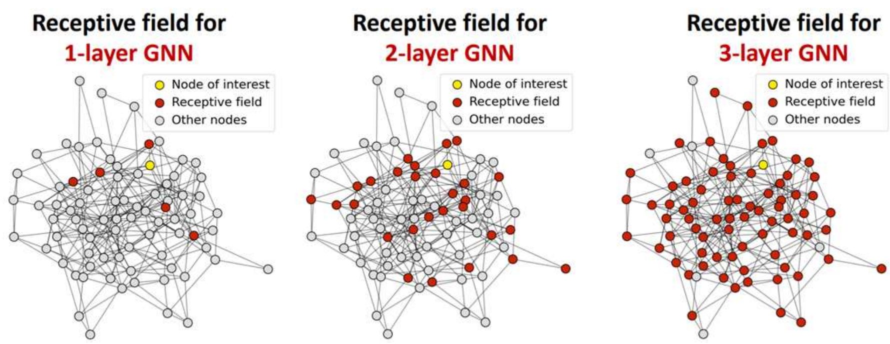


Acknowledgement: Prof. Jure Leskovec

Over-Smoothing Problem

47

- All the node embeddings converge to the same value
  - Over-smoothing can be explained via the notion of the receptive field
    - **Receptive field:** the set of nodes that determine the embedding of a node of interest
  - If two nodes have highly-overlapped receptive fields, then their embeddings are highly similar

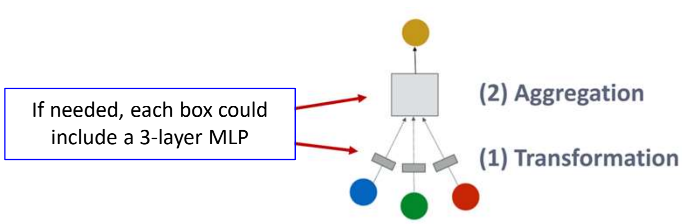


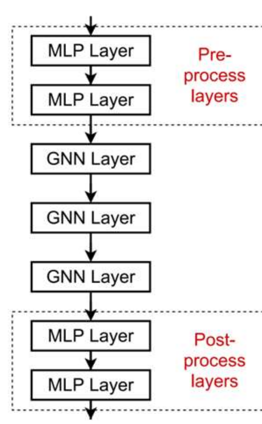
Acknowledgement: Prof. Jure Leskovec

Over-Smoothing Problem

48

- **Do not set GNN layers to be unnecessarily large!**
  - **Step 1:** Analyze the necessary receptive field to solve your problem
  - **Step 2:** Set number of GNN layers L to be a bit more than the receptive field we like.
- **How to enhance the expressive power of a GNN, if the number of GNN layers is small?**
  - **Solution 1:** Increase the expressive power within each GNN layer
  - **Solution 2:** Add layers that do not pass messages





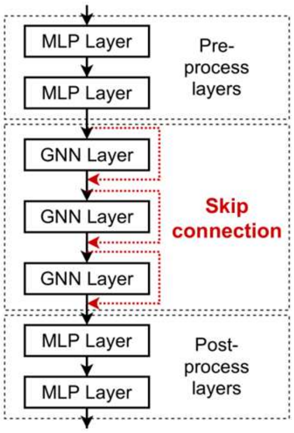
Acknowledgement: Prof. Jure Leskovec

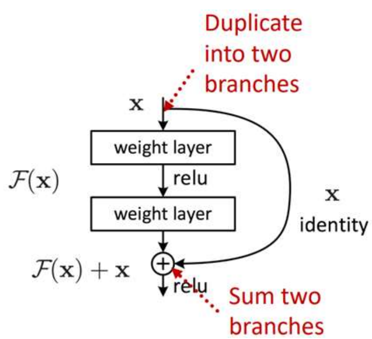
Over-Smoothing Problem

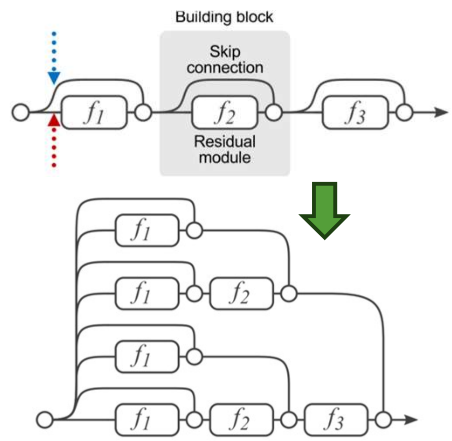
49

- **What if we still need many GNN layers?**
  - Add skip connections in GNNs

N skip connections → 2^N possible paths







Acknowledgement: Prof. Jure Leskovec

GCN with Skip Connection

50

- A standard GCN layer

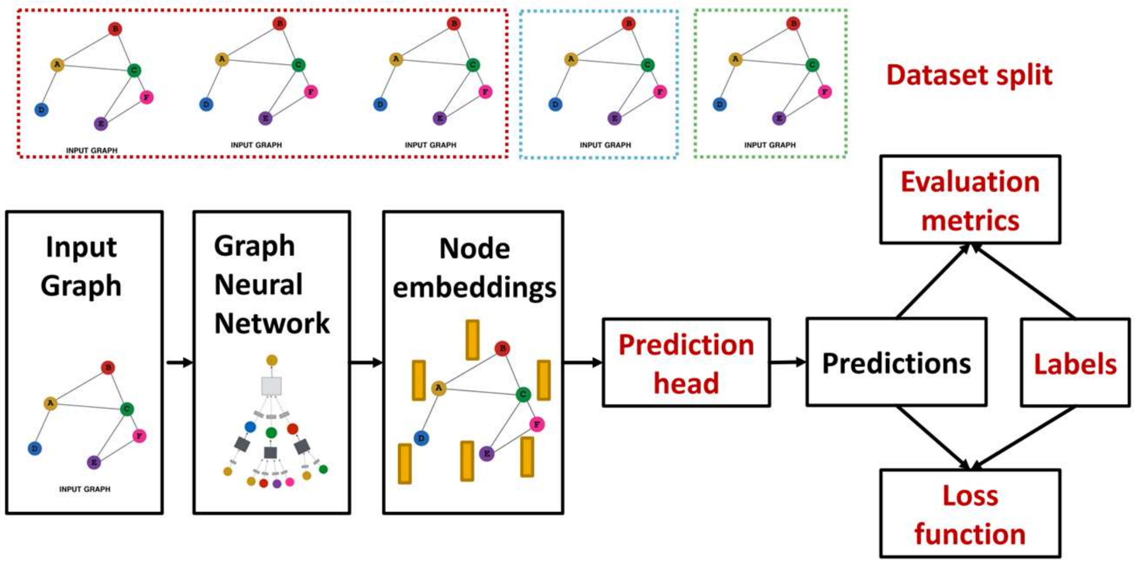
$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \mathbf{W}^{(l)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right)$$

- A GCN layer with skip connection

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \mathbf{W}^{(l)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|} + \mathbf{h}_v^{(l-1)}\right)$$

Acknowledgement: Prof. Jure Leskovec

GNN Training



Acknowledgement: Prof. Jure Leskovec

GNN Training: Prediction Head

- **Node-level prediction:**

$$\hat{\boldsymbol{y}}_v = \mathrm{Head}_{\mathrm{node}}(\mathbf{h}_v^{(L)}) = \mathbf{W}^{(H)}\mathbf{h}_v^{(L)}$$

  - $\mathbf{W}^{(H)} \in \mathbb{R}^{k \times d}$: We map node embeddings from $\mathbf{h}_v^{(L)} \in \mathbb{R}^d$ to $\hat{y}_v \in \mathbb{R}^k$ so that we can compute the loss

- **Edge-level prediction:**

$$\hat{\boldsymbol{y}}_{\boldsymbol{uv}} = \mathrm{Head}_{\mathrm{edge}}(\mathbf{h}_u^{(L)}, \mathbf{h}_v^{(L)})$$

$$\hat{\boldsymbol{y}}_{\boldsymbol{uv}} = \mathrm{Linear}(\mathrm{Concat}(\mathbf{h}_u^{(L)}, \mathbf{h}_v^{(L)}))$$

$$\hat{\boldsymbol{y}}_{\boldsymbol{uv}} = (\mathbf{h}_u^{(L)})^T \mathbf{h}_v^{(L)}$$

- **Graph-level prediction:**

$$\hat{\boldsymbol{y}}_G = \mathrm{Head}_{\mathrm{graph}}(\{\mathbf{h}_v^{(L)} \in \mathbb{R}^d, \forall v \in G\})$$

  - **(1) Global mean pooling**
    $$\hat{\boldsymbol{y}}_G = \mathrm{Mean}(\{\mathbf{h}_v^{(L)} \in \mathbb{R}^d, \forall v \in G\})$$
  - **(2) Global max pooling**
    $$\hat{\boldsymbol{y}}_G = \mathrm{Max}(\{\mathbf{h}_v^{(L)} \in \mathbb{R}^d, \forall v \in G\})$$
  - **(3) Global sum pooling**
    $$\hat{\boldsymbol{y}}_G = \mathrm{Sum}(\{\mathbf{h}_v^{(L)} \in \mathbb{R}^d, \forall v \in G\})$$

Acknowledgement: Prof. Jure Leskovec

GNN Training: Ground-Truth

53

- Supervised:
  - **Node labels $\boldsymbol{y}_v$:** in a citation network, which subject area does a node belong to
  - **Edge labels $\boldsymbol{y}_{uv}$:** in a transaction network, whether an edge is fraudulent
  - **Graph labels $\boldsymbol{y}_G$:** among molecular graphs, the drug likeness of graphs
- Unsupervised:
  - **Node-level $\boldsymbol{y}_v$.** Node statistics: such as clustering coefficient, PageRank, …
  - **Edge-level $\boldsymbol{y}_{uv}$.** Link prediction: hide the edge between two nodes, predict if there should be a link
  - **Graph-level $\boldsymbol{y}_G$.** Graph statistics: for example, predict if two graphs are isomorphic

Acknowledgement: Prof. Jure Leskovec

GNN Training: Loss

54

- **Classification:** Cross Entropy Loss

$$\mathrm{CE}\big(\underbrace{\boldsymbol{y}^{(i)}}_{\text{Label}},\underbrace{\widehat{\boldsymbol{y}}^{(i)}}_{\text{Prediction}}\big) = -\sum_{j=1}^{K} \boldsymbol{y}_j^{(i)} \log(\widehat{\boldsymbol{y}}_j^{(i)})$$

i-th data point

j-th class

$$\text{Loss} = \sum_{i=1}^{N} \mathrm{CE}\big(\boldsymbol{y}^{(i)},\widehat{\boldsymbol{y}}^{(i)}\big)$$

- **Regression:** Mean Squared Loss

$$\mathrm{MSE}\big(\boldsymbol{y}^{(i)},\widehat{\boldsymbol{y}}^{(i)}\big) = \sum_{j=1}^{K} (\boldsymbol{y}_j^{(i)} - \widehat{\boldsymbol{y}}_j^{(i)})^2$$

i-th data point

j-th target

$$\text{Loss} = \sum_{i=1}^{N} \mathrm{MSE}\big(\boldsymbol{y}^{(i)},\widehat{\boldsymbol{y}}^{(i)}\big)$$

Acknowledgement: Prof. Jure Leskovec

GNN Training: Evaluation Metrics

55

- Regression
  - Root mean square error (RMSE)

$$\sqrt{\sum_{i=1}^{N} \frac{(\boldsymbol{y}^{(i)} - \widehat{\boldsymbol{y}}^{(i)})^2}{N}}$$

  - Mean absolute error (MAE)

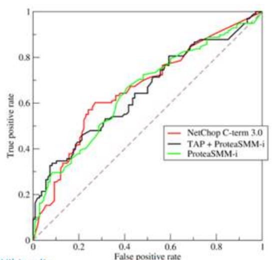
$$\frac{\sum_{i=1}^{N} \left|\boldsymbol{y}^{(i)} - \widehat{\boldsymbol{y}}^{(i)}\right|}{N}$$

- Classification
  - (1) Multi-class classification
    - We simply report the accuracy

$$\frac{1\left[\operatorname{argmax}\left(\widehat{\boldsymbol{y}}^{(i)}\right) = \boldsymbol{y}^{(i)}\right]}{N}$$

  - (2) Binary classification
    - Metrics sensitive to classification threshold
      - Accuracy
      - Precision / Recall
      - If the range of prediction is [0,1], we will use 0.5 as threshold
    - Metric Agnostic to classification threshold
      - ROC AUC

TPR



age Credit: Wikipedia

FPR

$$\mathrm{TPR} = \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

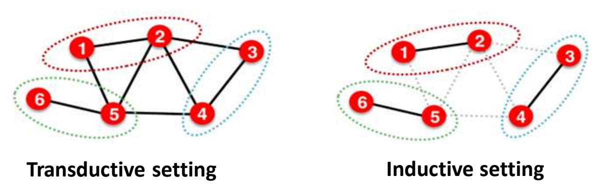
Acknowledgement: Prof. Jure Leskovec

GNN Training: Dataset Split

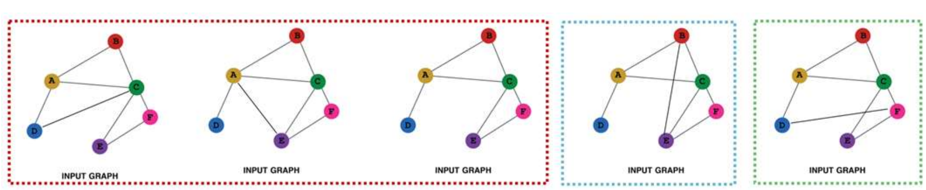
56

- Node classification:

Training
Validation
Test



- Graph Classification:

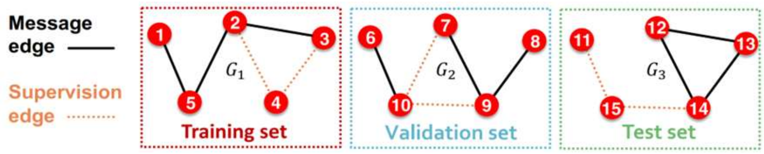


Acknowledgement: Prof. Jure Leskovec

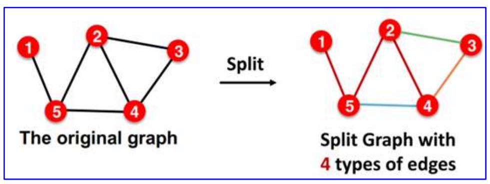
GNN Training: Dataset Split

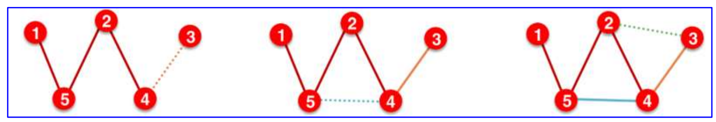
57

- Link Prediction:



Inductive setting



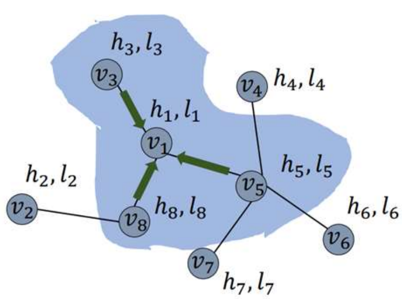


Transductive setting

Acknowledgement: Prof. Jure Leskovec

Spatial based Filtering

58



h_i: The hidden features

l_i: The input features

$$h_i^{(k+1)} = \sum_{v_j \in N(v_i)} f\left(l_i, h_j^{(k)}, l_j\right), \qquad \forall\, v_i \in V.$$

$N(v_i)$: Neighbors of the node v_i.

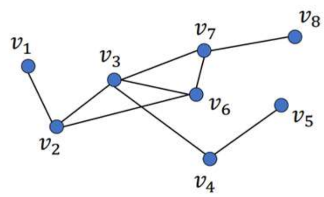
$f(\cdot)$: Feedforward neural network.

Ack: Yiqi Wang, Wei Jin, Yao Ma, and Jiliang Tang

Spectral based Filtering

59

- "***Spectral***" in the graph domain denotes the eigendecomposition of the graph Laplacian matrix into simpler orthonormal basis components



Adjacency Matrix: $A[i,j] = 1$ if v_i is adjacent to v_j
$A[i,j] = 0$, otherwise

Degree Matrix: $\mathbf{D} = \mathrm{diag}(degree(v_1), \ldots, degree(v_N))$

$$\underset{\boldsymbol{D}}{\underbrace{\begin{pmatrix}1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0\\0&0&4&0&0&0&0&0\\0&0&0&2&0&0&0&0\\0&0&0&0&2&0&0&0\\0&0&0&0&0&4&0&0\\0&0&0&0&0&0&3&0\\0&0&0&0&0&0&0&1\end{pmatrix}}_{\text{Degree Matrix}}} - \underset{\boldsymbol{A}}{\underbrace{\begin{pmatrix}0&1&0&0&0&0&0&0\\1&0&1&0&0&1&0&0\\0&1&0&1&0&1&1&0\\0&0&1&0&1&0&0&0\\0&0&0&1&0&1&0&0\\0&1&1&0&1&0&1&0\\0&0&1&0&0&1&0&1\\0&0&0&0&0&0&1&0\end{pmatrix}}_{\text{Adjacency Matrix}}} = \underset{\boldsymbol{L}}{\underbrace{\begin{pmatrix}1&-1&0&0&0&0&0&0\\-1&3&-1&0&0&-1&0&0\\0&-1&4&-1&0&-1&-1&0\\0&0&-1&2&-1&0&0&0\\0&0&0&-1&2&-1&0&0\\0&-1&-1&0&-1&4&-1&0\\0&0&-1&0&0&-1&3&-1\\0&0&0&0&0&0&-1&1\end{pmatrix}}_{\text{Laplacian Matrix}}}$$

Ack: Yiqi Wang, Wei Jin, Yao Ma, and Jiliang Tang

Spectral based Filtering

60

Laplacian matrix has a complete set of orthonormal eigenvectors:

$$\mathbf{L} = \underset{\boldsymbol{U}}{\begin{bmatrix} | & & | \\ \mathbf{u}_0 & \cdots & \mathbf{u}_{N-1} \\ | & & | \end{bmatrix}} \underset{\boldsymbol{\Lambda}}{\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix}} \underset{\boldsymbol{U^T}}{\begin{bmatrix} — & \mathbf{u}_0 & — \\ & \vdots & \\ — & \mathbf{u}_{N-1} & — \end{bmatrix}}$$

Eigenvalues are sorted non-decreasingly:

$$0 = \lambda_0 < \lambda_1 \leq \cdots \lambda_{N-1}$$

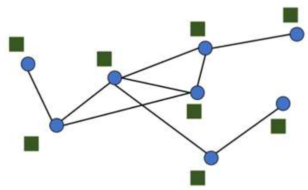
The frequency of an eigenvector of Laplacian matrix is its corresponding eigenvalue

Ack: Yiqi Wang, Wei Jin, Yao Ma, and Jiliang Tang

Spectral based Filtering

61

- Graphs and Graph Signals



Graph Signal: $f : \mathcal{V} \rightarrow \mathbb{R}^N$

$$\mathcal{V} = \{v_1, \ldots, v_N\}$$

$$\mathcal{E} = \{e_1, \ldots, e_M\}$$

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

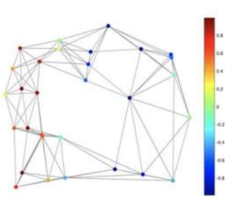
$$\mathcal{V} \longrightarrow \begin{bmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{bmatrix}$$

Ack: Yiqi Wang, Wei Jin, Yao Ma, and Jiliang Tang

Spectral based Filtering

62

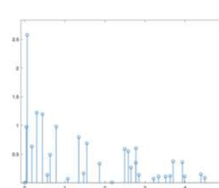
- Graph Fourier Transform (GFT)



Spatial domain: $\boldsymbol{f}$

$$\hat{\boldsymbol{f}} = \boldsymbol{U}^T \boldsymbol{f}$$

Decompose signal f



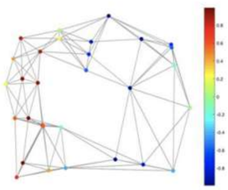
Spectral domain: $\hat{\boldsymbol{f}}$

Ack: Yiqi Wang, Wei Jin, Yao Ma, and Jiliang Tang

Spectral based Filtering

63

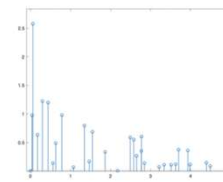
- Inverse Graph Fourier Transform (IGFT)



Spatial domain: $\boldsymbol{f}$

$\boldsymbol{f} = \boldsymbol{U}\hat{\boldsymbol{f}}$

Reconstruct signal $\boldsymbol{f}$



Spectral domain: $\hat{\boldsymbol{f}}$

Ack: Yiqi Wang, Wei Jin, Yao Ma, and Jiliang Tang

Spectral based Filtering

64

Filter a graph signal f:

$f \xrightarrow[\text{Decompose}]{GFT} U^T f \xrightarrow[\text{Filter}]{\hat{g}(\Lambda)} \hat{g}(\Lambda)U^T f \xrightarrow[\text{Reconstruct}]{IGFT} U\hat{g}(\Lambda)U^T f$

Coefficients

Filtered coefficients

$$\hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{N-1}) \end{bmatrix}$$

$f \xrightarrow[\text{Filtering}]{} U\hat{g}(\Lambda)U^T f$

$\hat{g}(L)$

Ack: Yiqi Wang, Wei Jin, Yao Ma, and Jiliang Tang

Questions?