# Graph 4. Node Embedding

- Laplacian Eigenmaps (NIPS 2002)
- Random Walk (KDD 2014)

1

#### Lectures on Graph-based ML

✓ Graph 1-3. Social network analysis (e.g., HITS & PageRank)

- Graph 4. Node embedding (Laplacian eigenmaps vs. random walk methods)
- Graph 5-6. Graph neural networks (GCN, GAT, GIN, etc.)
- Graph 7-8. Knowledge graph embedding
- Graph 9-11. Neural solvers for combinatorial optimization (AR, NAR, LLM's)
- Graph 12-13. Graph-based learning for recommender systems (invited talks)
- Graph 14-15. Learning with heterogeneous graphs

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

#### Motivation for Graph Node Embedding

- GloVe co-occurrence graph → word embedding → NLP
- Hyperlinked websites → page embedding → websites classification
- Citation graphs → article embedding → literature classification
- Co-author graphs → author embedding → community detection
- Molecular structure graph → atom embedding → Al for science
- •

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

3

3

#### **Unified View**

- Nodes can be any objects (words, documents, authors, atoms, etc.)
- Links represent the interactions or dependencies among nodes.
- Embedding Vectors
  - o Capturing the latent features of nodes based on graph structures
  - Supporting down-stream prediction tasks (node/graph classification, community detection, dense retrieval, etc.)

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

## Node Embedding Methods

- Based on linked structures only (this lecture)
  - Matrix-factorization based methods (e.g., Laplacian Eigenmaps)
  - Random-walk based methods
- Based on both linked structures and node-specific features (next lecture)

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

5

5

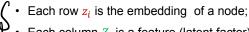
#### Input and Output

- Input Graph G = (V, A)
  - *V* is the set of *n* nodes in the graph;
  - $A \in \mathbb{R}^{n \times n}$  is the adjacency matrix with  $A_{ij} \geq 0$  and  $A_{ii} = 0$  (zero at the diagonal).



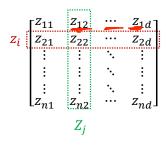
• Specifically, we focus on undirected graphs with  $A_{ij} = A_{ji}$  (symmetric)

• Output Matrix  $Z \in \mathbb{R}^{n \times d}$  (d < n)



• Each column  $Z_j$  is a feature (latent factor) of the embedding space.

Matrix Z



3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

6

#### **Graph Laplacian Matrices**

Default Version

$$L\stackrel{\text{def}}{=} D-A$$
 with  $D\stackrel{\text{def}}{=} diag(D_{11},...,D_{nn})$  and  $D_{ii}=\sum_j A_{ij}$ 

Symmetric Version

$$L_{sym} \stackrel{\text{def}}{=} D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = D^{-\frac{1}{2}} D D^{-\frac{1}{2}} - D^{-\frac{1}{2}} D^{-\frac{1}{2}} - D^{-\frac{1}{2}} D^{-\frac{1}{2}}$$
 
$$\left( A_{sym}[i,j] = \frac{A_{ij}}{\sqrt{D_{ii}}\sqrt{D_{ij}}} \right)$$

Random-walk Version

$$L_{rw} \stackrel{\text{def}}{=} D^{-1}L = \overbrace{D^{-1}D}^{I_{n\times n}} - \overbrace{D^{-1}A}^{A_{rm}} \qquad \qquad \left( \underbrace{A_{rm}[i,j] = \frac{A_{ij}}{D_{ii}}, \ \sum_{j} A_{rm}[i,j] = 1} \right)$$

3/14/2024

riming Yang, 11-741 Lecture on Graph-based Node Embedding

7

#### Properties of Graph Laplacian L

(Von Luxburg, 2007, https://arxiv.org/pdf/0711.0189.pdf)

- When A is symmetric, L = D A is also symmetric (obvious).
- L allows us to reinforce the smoothness of node embedding.
- *L* is positive semi-definite.  $2^{\mathsf{T}} \; \bigcup \; \mathcal{Z}$



L has n real-valued non-negative eigenvalues.

 $\lambda_1$  =0 is the smallest eigenvalue of L, and  $\mathbf{1}_n$  as the corresponding eigenvector.

L has a complete set of n eigenvectors (proof omitted).

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

# Laplacian eigenmaps and spectral techniques for embedding and clustering [M. Belkin and P. Niyogi. NIPS 2002]

**Task**: Given graph G = (V, A), find the optimal solution

$$Z^* = arg \min_{Z \in \mathbb{R}^{n \times d}} \left| z_i - z_j \right|^2 A_{i,j} \quad \leftarrow \text{Smoothness Penalty}$$

where the row vectors are node embeddings (d-dimensional vectors).

Optimal Solution

$$Z^* = (u_2, u_3, ..., u_{d+1})$$



where  $u_2,\ u_3,\ldots,u_{d+1}$  are the eigenvectors of the graph Laplacian (whichever the version of  $L,L_{sym}$  or  $L_{rm}$ ) sorted by the eigenvalues  $0<\lambda_2\leq\lambda_3\leq\cdots\leq\lambda_{k+1}$ .

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

9

# Smoothness Penalty (when d = 1)

• Let us consider d=1 for now, i.e.,  $z_i, z_j \in \mathbb{R}$  (scalers) and  $Z \in \mathbb{R}^n$  (vector)

$$\begin{split} \sum_{i,j} |z_i - z_j|^2 A_{i,j} &= \sum_{i,j} (z_i - z_j)^2 A_{ij} \\ &= \sum_{i,j} (z_i^2 + z_j^2 - 2z_i z_j) A_{ij} \\ &= \sum_i z_i^2 \underbrace{\sum_j A_{ij}}_{D_{ii}} + \sum_j z_j^2 \underbrace{\sum_i A_{ij}}_{D_{jj}} - \sum_{ij} 2z_i z_j A_{ij} \\ &= \underbrace{2\sum_i z_i^2 D_{ii} - 2\sum_{ij} z_i z_j A_{ij}}_{= 2Z^T LZ} \end{split}$$

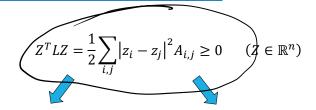
Also, we have

$$Z^{T}LZ = \sum_{ij} z_i z_j \frac{L_{ij}}{L_{ij}} = \sum_{i,j} z_i z_j \left( \frac{D_{ij} - A_{ij}}{D_{ij} - A_{ij}} \right) = \sum_{i} z_i^2 D_{ii} - \sum_{ij} z_i z_j A_{ij}$$

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

### Objective for Optimization (with d = 1)



$$\min_{Z \in \mathbb{R}^n} \sum_{i,j} |z_i - z_j|^2 A_{i,j} = \min_{Z \in \mathbb{R}^n} Z^T LZ$$

L is positive-semidefinite

 $(:Z^TLZ \ge 0 \text{ for any } Z \in \mathbb{R}^n).$ 

Smoothness Penalty

3/14/2024

11

#### Being positive-semidefinite (PSD) of L ...



- For any  $x \in \mathbb{R}^n$ , we have  $x^T L x \ge 0$ , according to the definition of a PSD matrix.
  - 2) For any eigenvector  $u_i \in \mathbb{R}^n$  of graph Laplacian L, we have  $u_i^T L u_i \geq 0$ .
  - 3)  $Lu_i = \lambda_i u_i \xrightarrow{multiplying \ u_i^T \ on \ both \ sides} u_i^T Lu_i = \lambda_i \xrightarrow{imply} \lambda_i \ge 0, \forall i \ (by \ Item 2 \ above)$





- 4)  $\lambda_1 = 0$  is the smallest eigenvalue of L.
- 5)  $u_1 = 1_n$  is the eigenvector corresponding to  $\lambda_1 = 0$  (next slide).
- 6) Vector  $1_n = Z^* = arg \min_{Z \in \mathbb{R}^n} Z^T L Z$  is naively optimal or 1D embedding.



We want to modify our objective as  $Z^* = arg\min_{Z \perp u_1} Z^T L Z$ , which leads to  $u_2$  with  $\lambda_2 > 0$ .

#### Eigenvector $u_1 = 1_n$ with $\lambda_1 = 0$

• Starting with L = D - A, we have

$$L\mathbf{1}_{n} = D\mathbf{1}_{n} - A\mathbf{1}_{n} \qquad \text{(multiplying } \mathbf{1}_{n} \text{ on both sides)}$$

$$= \begin{pmatrix} D_{11} \\ D_{22} \\ \vdots \end{pmatrix} - \begin{pmatrix} \sum_{j=1}^{n} A_{1j} \\ \sum_{j=1}^{n} A_{2j} \\ \vdots \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ 0 \\ \vdots \end{pmatrix} = 0 \cdot \mathbf{1}_{n}$$

• Thus, we have  $L\mathbf{1}_n = 0 \cdot \mathbf{1}_n$  which means  $\mathbf{u}_1 = \mathbf{1}_n$  and  $\lambda_1 = 0$ .

3/14/2024

Yiming Yang, 11-741 Lecture on Graph-based Node Embeddin

13

13

#### Smoothness penalty when d > 1

$$\begin{split} \sum_{i,j} & |z_i - z_j|^2 A_{i,j} &= \sum_{i,j} \sum_{k=1}^d \left( z_i^{(k)} - z_j^{(k)} \right)^2 A_{i,j} & \left( z_i, z_j \in \mathbb{R}^d \right) \\ &= \sum_{k=1}^d \sum_{i,j} \left( z_i^{(k)} - z_j^{(k)} \right)^2 A_{i,j} \\ &= \sum_{k=1}^d \underbrace{2Z_k}^T L Z_k & \text{(using the proof for } d = 1) \\ & \stackrel{?}{=} 2tr(Z^T L Z) & \text{(next slide)} \end{split}$$



$$\min_{Z \in \mathbb{R}^{n \times d}} \sum_{i,j} \left| z_i - z_j \right|^2 A_{i,j} = \min_{Z \in \mathbb{R}^{n \times d}} tr(Z^T L Z) \qquad (d \ge 1)$$

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

#### What is the trace of matrix $Z^T LZ$ ?

• Let  $(Z_1, Z_2, ..., Z_d)$  be the column vectors of matrix  $Z \in \mathbb{R}^{n \times d}$ .

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

15

Optimal Embedding

• We want: 
$$Z^* = \underset{Z_k \perp 1_n}{argmin} \sum_{k=1}^d Z_k^T L Z_k$$

• When  $d = 1$ ,  $Z^* = \underset{Z_1 \perp 1_n}{argmin} Z_1^T L Z_1 = u_2$ 

• When  $d > 1$ , for  $k = 1$  to  $d + 1$ , find

$$Z^*_{k+1} = \underset{Z_{k+1} \perp \{u_1, \dots, u_k\}}{argmin} Z_k^T L Z_k = u_{k+1}$$

return  $Z^* = (u_2, \dots, u_{d+1})$ 

 $\operatorname{tr}(Z^{*T}LZ^{*}) = \sum_{k=2}^{d+1} \mathbf{u}_{k}^{T} L \mathbf{u}_{k} = \lambda_{2} + \lambda_{3} + \dots + \lambda_{d+1}$   $\text{hoose } d? \quad \text{for all } \mathbf{v} \text{ for all }$ • How do we choose d?

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

#### Generalized Eigenvector Problem

[M. Belkin and P. Niyogi. NIPS 2002]

Standard eigenvector problem is defined as to find all the vectors satisfying

$$L\mathbf{u} = \lambda \mathbf{u}$$

or 
$$\mathbf{u}^T L \mathbf{u} = \lambda$$

- Solution: eigenvectors  $u_1, u_2, ... u_n$  ordered by  $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ .
- Generalized eigenvector problem is defined as to find all the vectors satisfying

$$L\mathbf{v} = \lambda \mathbf{D}\mathbf{v}$$

$$\mathbf{v}^T L \mathbf{v} = \lambda$$

 $\mathbf{y}^T L \mathbf{y} = \lambda$  (D is a diagonal matrix.)

Solution: D-orthonormal  $y_1, y_2, ... y_n$  (when D is splitable)

$$\forall_{i,j} \mathbf{y}_i^T \mathbf{D} \mathbf{y}_j = \begin{cases} 1 & if \ i = j \\ 0 & if \ i \neq j \end{cases} \quad \text{or} \quad Y^T D Y = I, \ Y = (\mathbf{y}_1, \mathbf{y}_2, \dots \mathbf{y}_n)$$

3/14/2024

This is because the dot product of a whit ledor with itself is

#### Other matrix-factorization based methods

Laplacian Eigenmaps (M. Belkin and P. Niyogi. NIPS 2002)

$$Z^* = arg\min_{Z_{n\times d}} |z_i - z_j|^2 A_{i,j} .$$

Graph Factorization (A. Ahmed et al., WWW 2013)

$$Z^* = arg\min_{Z_{n\times d}} \sum_{i,j} |z_i^T z_j - A_{i,j}|$$

GraRep (S. Cao et al., CIKM 2015)

$$\min_{orthononal Z^k} \sum_{i,j} \left\| z_i^T z_j - \log \left( \frac{A_{i,j}^k}{\sum_l A_{l,j}^k} \right) + \log \frac{\lambda}{n} \right\|_2^2$$

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

#### Issue with the computational cost



Eigen-decomposition of the full matrix is expensive when n is large.

A chapter alternative is the random walk approach (next).

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embeddin

19

19

# Node Embedding via Random Walk over a Graph

(e.g., DeepWalk by Perozzi et al., KDD 2014)

- Input graphs G = (V, E), for example
  - BlogCatalog: a network of social relationships among blogger authors (10k)
  - Flickr: a network of the contacts between users (80k) on a photo sharing website
  - YouTube: a social network among users (1,139k) of a popular video sharing website
- Random walk for generating "word"/context pairs
  - From each node, randomly walk for t steps to obtain a sequence of nodes ("words")
  - · At each step, uniformly sample the next node from the neighbors of the current node
  - · Apply a sliding window over the node sequences to obtain "word"/context pairs
  - Train a word2vec method (SkipGram) on the above training pairs to obtain the embedding of nodes

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

20

#### Other Variants of Random Walk Methods

- DeepWalk used a uniform sampling strategy, but other teleportation strategies are also possible (A Grover & J Leskovec, KDD 2016)
  - To reduce the probability of turning back to each node
  - To reduce the link weights for the nodes which have many links
  - To sample the context of each node via beam search (sampling multiple nodes per step instead sampling one node per step)
- DeepWalk uses SkipGram as the word embedding algorithm, but other choices (CBOW, GloVe, etc.) are also possible.

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

2

21

### **Concluding Remarks**

- Matrix-factorization methods focus on the global smoothness of the entire graph (the eigendecomposition of the graph Laplacian matrix).
- Random-walk methods focus on the local neighborhood of each node.
- ☐ The former could be too expensive for large graphs.
- The latter could be too simple for good performance.
  - Both methods are task-agnostic, which is a limitation as the embeddings of nodes cannot be adapted to down-stream tasks.
  - Both do not leverage node-specific features (even if available) as another weakness.
  - Graph Neural Networks (GNNs) overcome both kinds of the limitations (next lecture).

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

#### References

- Von Luxburg, Ulrike. "A tutorial on spectral clustering." Statistics and computing 17.4 (2007).
- M. Belkin and P. Niyogi. "Laplacian eigenmaps and spectral techniques for embedding and clustering", NIPS 2002.
- Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." KDD 2014.
- A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, A. Smola. Distribute Large-scale Natural Graph Factorization. WWW 2013.
- Cao, Shaosheng, Wei Lu, and Qiongkai Xu. "Grarep: Learning graph representations with global structural information." CIKM 2015.
- Hamilton, William L., Rex Ying, and Jure Leskovec. "Representation learning on graphs: Methods and applications." arXiv preprint arXiv:1709.05584 (2017).
- Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." KDD 2016

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding

23

23

#### Graph construction based on given node features

(M. Belkin and P. Niyogi. NIPS 2002)

- Let  $x_1, x_2, \dots, x_n \in \mathbb{R}^l$  be the given feature vectors of nodes, we can construct a graph based on
  - $\circ$  ε-neighborhoods ( $\varepsilon > 0$ ): connect nodes i and j if  $\left\|x_i x_j\right\|^2 < \varepsilon$ ;
  - o k-nearest neighbors ( $k \in \mathbb{N}$ ): connect nodes i to j if i is among the k-nearest neighbors of j or if j is among the k-nearest neighbors of i.
- Choices of link weights in adjacency matrices
  - Simple-minded:  $A_{ij} = 1$  if and only if nodes i and j are connected;
  - Heat kernel (with parameter t > 0):  $A_{ij} = e^{-\frac{\left\|x_i x_j\right\|^2}{t}}$  if nodes i and j are connected.

3/14/2024

@Yiming Yang, 11-741 Lecture on Graph-based Node Embedding