Summary Questions of the lecture

- Describe the key idea of APPNP: Approximated Personalized Propagation of Neural Prediction.

→ APPNP predicts (extracts) the node features using a shared neural network, then iteratively propagates the features(predictions) for K steps. The propagation is done by the random walk with the personalized teleport to the initial predictions, where the Chebyshev filter, which is also column stochastic, is used for random transition probability matrix.

$P = D^{-1}A$ ← *Ra.Walk*

$$\hat{\tilde{A}} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$$ ← *ChevNet*

$$Z^{(0)} = H = f_\theta(X),$$

$$Z^{(k+1)} = (1-\alpha)\hat{\tilde{A}}Z^{(k)} + \alpha H, k = 0, \ldots, K-2$$

$$Z^{(K)} = \mathrm{softmax}\left((1-\alpha)\hat{\tilde{A}}Z^{(K-1)} + \alpha H\right),$$

Summary Questions of the lecture

- What are the benefits of APPNP: Approximated Personalized Propagation of Neural Prediction.

→ After K steps of propagation, $Z^{(K)}$ becomes a weighted sum of $K, K-1, K-2, \cdots, 1$ –hop aggregations and the original node features H. The multiplicity of the Chebyshev filter makes the feature converge to the equilibrium point leads but can enlarge the smoothing region and so cause a over-smoothing. However, APPNP can prevent over-smoothing by the teleport to the original node features H and the attenuation of the propagation coefficients of high-order hops.

$$Z^{(K)} = (1-\alpha)\hat{\tilde{A}}Z^{(K-1)} + \alpha H$$

$$Z^{(K)} = (1-\alpha)\hat{\tilde{A}}\left((1-\alpha)\hat{\tilde{A}}Z^{(K-2)} + \alpha H\right) + \alpha H$$

$$Z^{(K)} = (1-\alpha)^2\hat{\tilde{A}}^2Z^{(K-2)} + (1-\alpha)\alpha\hat{\tilde{A}}H + \alpha H$$

$$Z^{(K)} = (1-\alpha)^K\hat{\tilde{A}}^K H + \cdots + (1-\alpha)\alpha\hat{\tilde{A}}H + \alpha H$$

Summary Questions of the lecture

- Discuss the difference among personalized PageRank, ShevNet, and APPNP.

→ The central differences lie in the type of filters (or transition matrices in random walk-sense) used, and whether explicitly retaining original features (or random teleports in random walk-sense) is allowed. Personalized PageRank uses the vanilla normalized adjacency matrix and allows teleports to a random query node. The simplified ChebNet uses the Chebyshev filter for the transition matrix, and does not allow the original features to be explicitly included. APPNP also uses the Chebyshev filter, but allows random teleports to the original features.

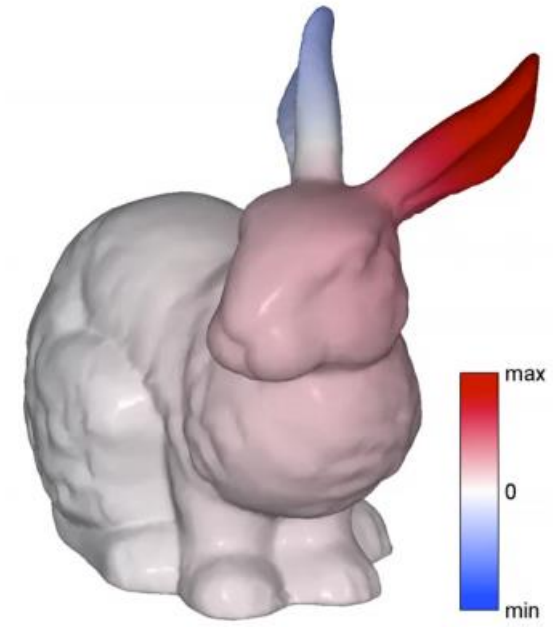
Outline of Lecture (5)

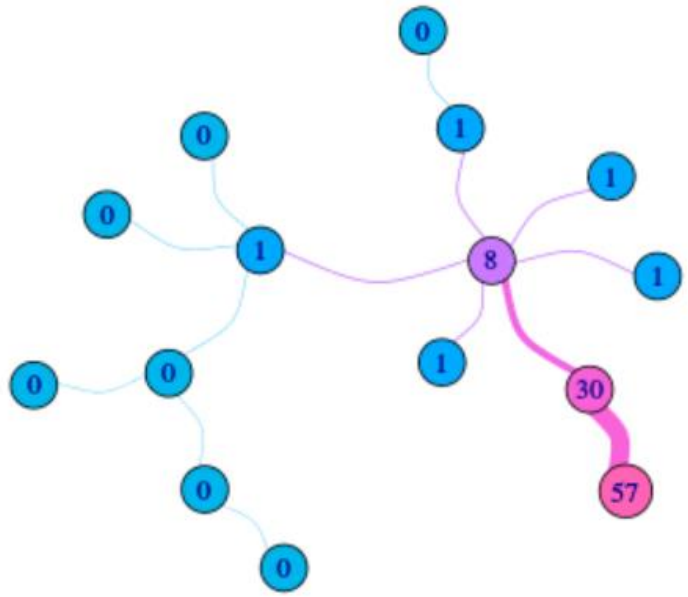
- Link Analysis
  - Directed Graph
    - Strongly Connected Graph
    - Directed Acyclic Graph
  - Link Analysis Algorithms
    - PageRank (Ranking of Nodes)
    - Random Teleports
    - Google Matrix
    - Sparse Matrix Formulation
    - Personalized PageRank
    - Random Walk with Restart

- Random Walks and Diffusion
- Diffusion in GCN
  - Propagation using graph diffusion
    - APPNP: Predict Then Propagate [ICLR'19]
    - Graph Diffusion-Embedding Networks [CVPR'19]
  - Making a new graph
    - Diffusion Improves Graph Learning [NIPS'19]
    - SSL with Graph Learning-Convolutional Networks [CVPR'19]

GCN:Graph Diffusion

(Continued) Random Walks and Diffusion, Diffusion in GCN





GCN: APPNP

- Predict Then Propagate: Graph Neural Networks Meet Personalized Pager ank [ICLR'19]
- PPNP: Personalized Propagation of Neural Prediction

$$\boldsymbol{Z_{PPNP}} = \mathrm{softamx}[\alpha\left(\mathbf{I} - (1-\alpha)\hat{\tilde{A}}\right)^{-1}\boldsymbol{H}]$$

- APPNP: Approximated Personalized Propagation of Neural Prediction

$$\hat{\tilde{\boldsymbol{A}}} = \tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}$$

$$\boldsymbol{Z}^{(0)} = \boldsymbol{H} = f_\theta(\boldsymbol{X}),$$

$$\boldsymbol{Z}^{(k+1)} = (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\boldsymbol{Z}^{(k)} + \alpha\boldsymbol{H}, k = 0, \ldots, K-2$$

$$\boldsymbol{Z}^{(K)} = \mathrm{softmax}\left((1-\alpha)\hat{\tilde{\boldsymbol{A}}}\boldsymbol{Z}^{(K-1)} + \alpha\boldsymbol{H}\right),$$

GCN: APPNP

- Predict Then Propagate: Graph Neural Networks Meet Personalized Pager ank [ICLR'19]

Table 2: Average accuracy with uncertainties showing the 95 % confidence level calculated by bootstrapping. Previously reported improvements vanish on our rigorous experimental setup, while PPNP and APPNP significantly outperform the compared models on all datasets.

Model	CITESEER	CORA-ML	PUBMED	MS ACADEMIC
V. GCN	73.51 ± 0.48	82.30 ± 0.34	77.65 ± 0.40	91.65 ± 0.09
GCN	75.40 ± 0.30	83.41 ± 0.39	78.68 ± 0.38	92.10 ± 0.08
N-GCN	74.25 ± 0.40	82.25 ± 0.30	77.43 ± 0.42	92.86 ± 0.11
GAT	75.39 ± 0.27	84.37 ± 0.24	77.76 ± 0.44	91.22 ± 0.07
JK	73.03 ± 0.47	82.69 ± 0.35	77.88 ± 0.38	91.71 ± 0.10
Bt. FP	73.55 ± 0.57	80.84 ± 0.97	72.94 ± 1.00	91.61 ± 0.24
PPNP*	$\mathbf{75.83 \pm 0.27}$	$\mathbf{85.29 \pm 0.25}$	-	-
APPNP	75.73 ± 0.30	85.09 ± 0.25	$\mathbf{79.73 \pm 0.31}$	$\mathbf{93.27 \pm 0.08}$

*out of memory on PUBMED, MS ACADEMIC (see efficiency analysis in Section 3)

Graph Diffusion (generalized)

- Generalized graph diffusion ⟵ $Z^{(K)} = (1-\alpha)^K \hat{\tilde{A}}^K H + \cdots + (1-\alpha)\alpha\hat{\tilde{A}}H + \alpha H$

  K –hop aggregation

  - P: Transition probability matrix
  - θ: weighting coefficient

$$S = \sum_{k=0}^{\infty} \theta_k P^k ,$$

$$P = AD^{-1} \; or \; D^{-1/2}AD^{-1/2} \; or \; (\mathrm{I}+D)^{-1/2}(\mathrm{I}+A)(\mathrm{I}+D)^{-\frac{1}{2}}$$

  - e.g. heat kernel & personalized PageRank

$$\theta_k^{HT} = e^{-t}\frac{t^k}{k!}, \qquad \theta_k^{HT} = \alpha(1-\alpha)^k$$

Graph Diffusion Convolution (GDC)

- Diffusion-Convolution Neural Networks [NIPS'16]
- Diffusion Improves Graph Learning [NeurIPS'19]
- Instead of aggregating information only from the first-hop neighbors, **Graph Diffusion Convolution (GDC)** aggregates information from a larger neighborhood (spatially localized)
- In practice, new graph is created via graph
  - Heat Kernel Diffusion
  - Personalized PageRank Diffusion
- $S = \sum_{k=0}^{\infty} \theta_k P^k$ becomes an adjacency matrix of a new graph, which improves the learning performance of a GCN.

Graph Diffusion Convolution (GDC)

- Generalized graph diffusion

$$S = \sum_{k=0}^{\infty} \theta_k P^k ,$$

- In general, S becomes a dense matrix, and so sparsification is done additionally for $\tilde{S}$
  - Top$-k$: Use the k entries with the highest mass per column, 2
  - Thresholding ϵ: Set entries below ϵ to zero
- Normalize (symmetric)

$$T_{sym}^{\tilde{S}} = D_{\tilde{S}}^{-1/2} \tilde{S} D_{\tilde{S}}^{-1/2}$$

- Apply other graph methods on $T_{sym}^{\tilde{S}}$

Graph Diffusion Convolution (GDC)

- Generalized graph diffusion

$$S = \sum_{k=0}^{\infty} \theta_k P^k ,$$

- Sparsification is done additionally for $\tilde{S}$
- Normalize (symmetric)

$$T_{sym}^{\tilde{S}} = D_{\tilde{S}}^{-1/2} \tilde{S} D_{\tilde{S}}^{-1/2}$$

- GCN (ShevNet): $\boldsymbol{F^l} = T_{sym}^{\tilde{S}} \boldsymbol{H^l}$, $\boldsymbol{H^{l+1}} = \boldsymbol{\sigma}(\boldsymbol{F^l \Theta})$
- GAT (Attention): $\boldsymbol{F^l} = T_{sym}^{\tilde{S}}(\boldsymbol{\alpha}) \boldsymbol{H^l}$, $\boldsymbol{H^{l+1}} = \boldsymbol{\sigma}(\boldsymbol{F^l \Theta})$
- Hyperparameters: θ_k (Heat kernels or APPNP), order of k

Graph Diffusion Convolution (GDC)

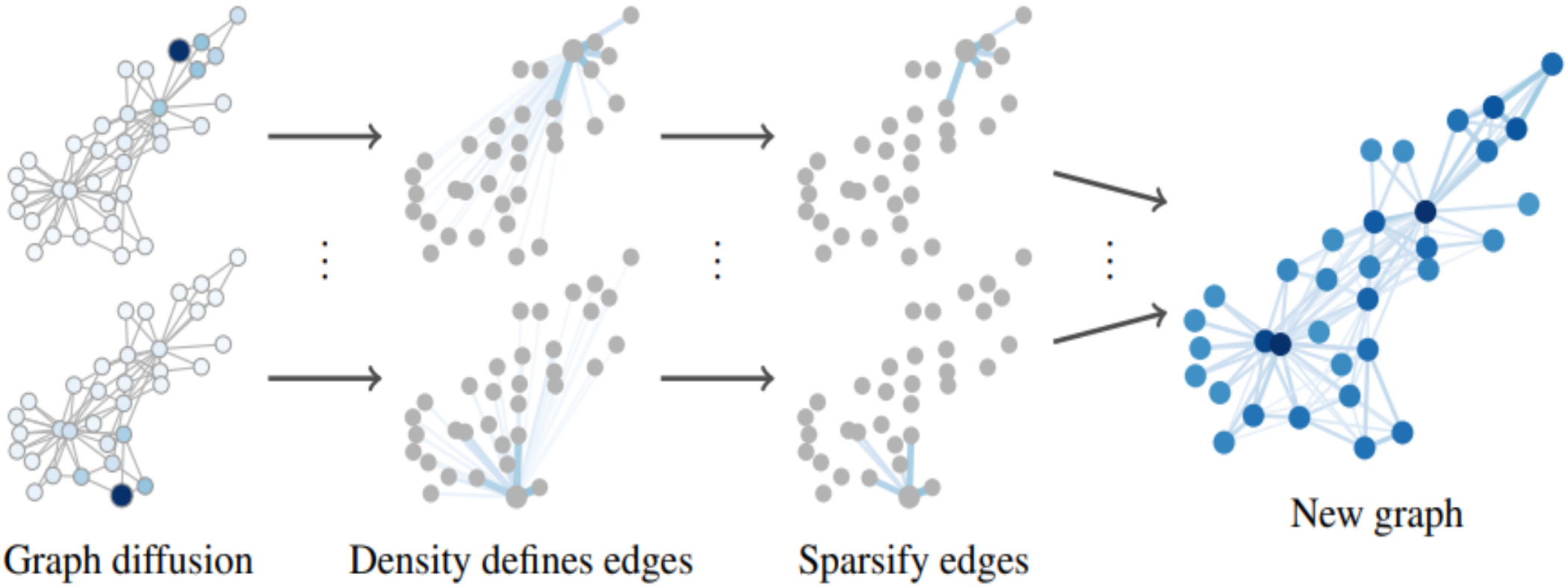


Figure 1: Illustration of graph diffusion convolution (GDC). We transform a graph $\boldsymbol{A}$ via graph diffusion and sparsification into a new graph $\tilde{\boldsymbol{S}}$ and run the given model on this graph instead.

Graph Diffusion Convolution (GDC)

Figure 3: Node classification accuracy of GNNs with and without GDC. GDC consistently improves accuracy across models and datasets. It is able to fix models whose accuracy otherwise breaks down.

Graph Diffusion-Embedding Networks **(GDEN)**

- Data Representation and Learning with Graph Diffusion-Embedding Networks [Bo Jiang et al., CVPR'19]
- Similar to GDC+GCN, but GDC is used as a aggregation function.
- Standard GCN:Standard GCN:

$$\mathbf{F}^{(l)} = (\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{H}^{(l)}$$

$$\mathbf{H}^{(k+1)} = \sigma\left(\mathbf{F}^{(l)}\,\mathbf{W}^{(l)}\right)$$

$$\hat{\tilde{A}} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$$

- This one-step diffusion does not return the equilibrium representation of feature diffusion which thus may lead to weak contextual feature representation.

Graph Diffusion-Embedding Networks **(GDEN)**

- Standard GCN: ShevNet

$$\mathbf{F}^{(l)} = (\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2})\mathbf{H}^{(l)}$$

$$\mathbf{H}^{(k+1)} = \sigma(\mathbf{F}^{(l)}\mathbf{W}^{(l)})$$

F: viewed as a feature diffusion method

- Graph Diffusion Convolution (GDC):

$$\mathbf{F}^{(l)} = (D_{\tilde{S}}^{-1/2}\tilde{S}D_{\tilde{S}}^{-1/2})\mathbf{H}^{(l)}$$

$$\tilde{S} \leftarrow S = \sum_{k=0}^{\infty}\theta_k P^k \leftarrow P = D^{-1}A$$

- Approx. Personalized Propagation of Neural Prediction (APPNP):

$$F^l = [(1-\alpha)^K \hat{\tilde{A}}^K + \sum_{k=0}^{K-1}(1-\alpha)^k \alpha \hat{\tilde{A}}^k]\, H^l$$

$$\hat{\tilde{A}} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$$

- PPNP:

$$F^l = \alpha(\mathbf{I} - (1-\alpha)\hat{\tilde{A}})^{-1}H^l$$

Graph Diffusion-Embedding Networks **(GDEN)**

- Graph Diffusion Convolution (GDC):

$$\mathbf{F}^{(l)} = (D_{\widetilde{S}}^{-1/2}\widetilde{S}D_{\widetilde{S}}^{-1/2})\mathbf{H}^{(l)}$$

$$\widetilde{S} \leftarrow S = \sum_{k=0}^{\infty} \theta_k P^k \leftarrow P = D^{-1}A$$

- PPNP:

$$F^l = \alpha(\mathbf{I} - (1-\alpha)\hat{\tilde{A}})^{-1}H^l$$

$$\hat{\tilde{A}} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$$

- Graph Diffusion-Embedding Network (GDEN):

Model	Diffusion operator $\mathcal{F}_d(\mathbf{A}, \mathbf{H})$
RWR Eq.(9)	$(1-\lambda)(\mathbf{I} - \lambda\mathbf{A}\mathbf{D}^{-1})^{-1}\mathbf{H}$
LapReg Eq.(11)	$\lambda(\mathbf{D} - \mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{H}$
NLapReg Eq.(13)	$(1-\gamma)(\mathbf{I} - \gamma\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})^{-1}\mathbf{H}$

Graph Diffusion-Embedding Networks **(GDEN)**

- Graph Diffusion-Embedding Network (GDEN):

Model	Diffusion operator $\mathcal{F}_d(\mathbf{A}, \mathbf{H})$
RWR Eq.(9)	$(1-\lambda)(\mathbf{I}-\lambda\mathbf{A}\mathbf{D}^{-1})^{-1}\mathbf{H}$
LapReg Eq.(11)	$\lambda(\mathbf{D}-\mathbf{A}+\lambda\mathbf{I})^{-1}\mathbf{H}$
NLapReg Eq.(13)	$(1-\gamma)(\mathbf{I}-\gamma\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})^{-1}\mathbf{H}$

- Random Walk with Restart (RWR)

$$\mathbf{f}_i^{(t+1)} = \lambda \sum\nolimits_{j,j\neq i} \mathbf{P}_{ij}\mathbf{f}_j^{(t)} + (1-\lambda)\mathbf{h}_i$$

- Laplacian Regularization

$$\min_{\mathbf{F}} \ \frac{1}{2}\sum_{i,j=1}^{n} \mathbf{A}_{ij}\|\mathbf{f}_i - \mathbf{f}_j\|_2^2 + \lambda\sum_{i=1}^{n}\|\mathbf{f}_i - \mathbf{h}_i\|_2^2$$

- Normalized Laplacian Regularization

$$\min_{\mathbf{F}} \ \frac{1}{2}\sum_{i,j=1}^{n} \mathbf{A}_{ij}\|\frac{\mathbf{f}_i}{\sqrt{\mathbf{d}_i}} - \frac{\mathbf{f}_j}{\sqrt{\mathbf{d}_j}}\|_2^2 + \lambda\sum_{i=1}^{n}\|\mathbf{f}_i - \mathbf{h}_i\|_2^2$$

Graph Diffusion-Embedding Networks **(GDEN)**

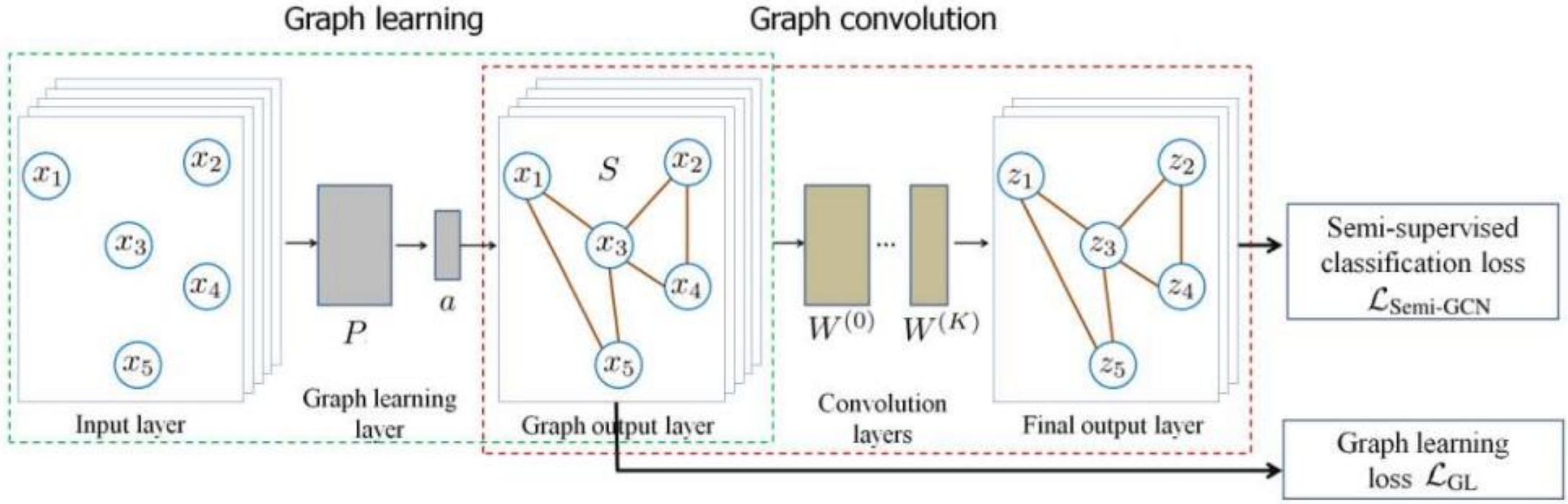
$$\mathbf{H}^{(k+1)} = \sigma\left(\mathcal{F}_d(\mathbf{A}, \mathbf{H}^{(k)})\mathbf{W}^{(k)}\right)$$

Table 2. Comparison results on citation network datasets

Methond	Citeseer	Cora	Pubmed
ManiReg [2]	60.1%	59.5%	70.7%
LP [36]	45.3%	68.0%	63.0%
DeepWalk [26]	43.2%	67.2%	65.3%
Planetoid [33]	64.7%	75.7%	77.2%
DCNN [1]	64.5%	76.7%	75.3%
GCN [15]	70.3%	83.6%	78.3%
GAT [30]	71.0%	83.2%	78.0%
GDEN-RWR	**72.8%**	82.0%	78.7%
GDEN-Lap	72.6%	84.7%	**78.9%**
GDEN-NLap	70.3%	**85.1%**	78.7%

Graph Learning-Convolutional Networks **(GLCN)**

- Semi-supervised Learning with Graph Learning-Convolutional Networks [Bo Jiang et al., CVPR'19]
- Key idea: Making a new graph then applying GCNs
- How to make a new graph? **Link prediction!**

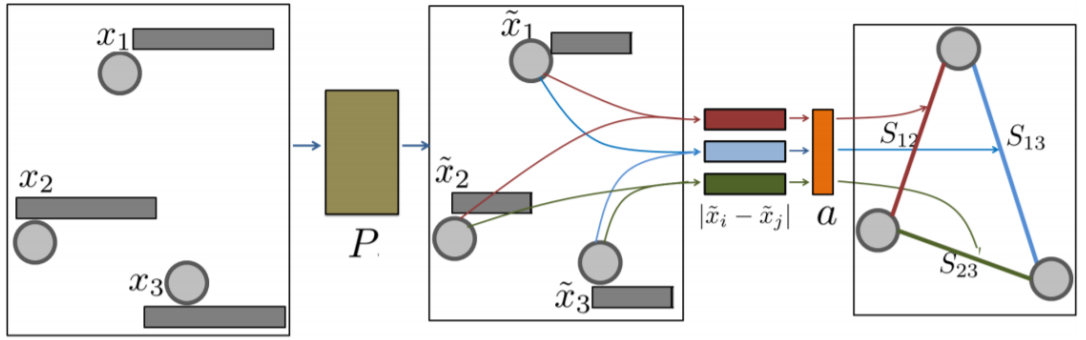


Graph Learning-Convolutional Networks **(GLCN)**

- Graph Learning Layer:
  - given an input: $X = (x_1, x_2 \cdots x_n) \in \mathbb{R}^{n \times p}$,
  - aim to seek a non-negative function $S_{ij} = g(x_i, x_j)$ for link prediction
  - implemented via a single-layer neural network, parameterized by a weight vector $a = (a_1, a_2, \cdots a_p)^T$

$$S_{ij} = g(x_i, x_j) = \frac{A_{ij} \exp(\mathrm{ReLU}(a^T |x_i - x_j|))}{\sum_{j=1}^{n} A_{ij} \exp(\mathrm{ReLU}(a^T |x_i - x_j|))}$$

(when A is not available, $A_{ij} = 1$)



- Graph Learning Loss:

$$\mathcal{L}_{\mathrm{GL}} = \sum_{i,j=1}^{n} \|x_i - x_j\|_2^2 S_{ij} + \gamma \|S\|_F^2 + \beta \|S - A\|_F^2$$

a is trained

Graph Learning-Convolutional Networks **(GLCN)**

- Total Loss:

$$\mathcal{L}_{\text{Semi-GLCN}} = \mathcal{L}_{\text{Semi-GCN}} + \lambda \mathcal{L}_{\text{GL}}$$

Table 1. Comparison results of semi-supervised learning on dataset Citeseer, Cora and Pubmed.

Methond	Citeseer	Cora	Pubmed
ManiReg [2]	60.1%	59.5%	70.7%
LP [23]	45.3%	68.0%	63.0%
DeepWalk [18]	43.2%	67.2%	65.3%
GCN [11]	70.9%	82.9%	77.9%
GAT [21]	71.0%	83.2%	78.0%
GLCN	**72.0%**	**85.5%**	**78.3%**

Summary: Diffusion in GCN

$$F^l = \alpha(\mathbf{I} - (1-\alpha)\hat{\tilde{A}})^{-1}H^l \qquad \hat{\tilde{A}} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$$

- Preliminaries:
  - Random Walk & Diffusion
  - PageRank

Model	Diffusion operator $\mathcal{F}_d(\mathbf{A}, \mathbf{H})$
RWR Eq.(9)	$(1-\lambda)(\mathbf{I} - \lambda\mathbf{A}\mathbf{D}^{-1})^{-1}\mathbf{H}$
LapReg Eq.(11)	$\lambda(\mathbf{D} - \mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{H}$
NLapReg Eq.(13)	$(1-\gamma)(\mathbf{I} - \gamma\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})^{-1}\mathbf{H}$

- Papers:
  - Propagation using graph diffusion
    - Predict Then Propagate: Graph Neural Networks Meet Personalized PageRank [ICLR'19]
    - Data Representation and Learning with Graph Diffusion-Embedding Networks [CVPR'19]
  - Making a new graph
    - Diffusion Improves Graph Learning [NIPS'19]
    - Semi-supervised Learning with Graph Learning-Convolutional Networks [CVPR'19]

$$\mathbf{F}^{(l)} = (D_{\tilde{S}}^{-1/2}\tilde{S}D_{\tilde{S}}^{-1/2})\mathbf{H}^{(l)} \qquad \tilde{S} \leftarrow S = \sum_{k=0}^{\infty}\theta_k P^k \leftarrow P = D^{-1}A$$

$$S_{ij} = g(x_i, x_j) = \frac{A_{ij}\exp(\mathrm{ReLU}(a^T|x_i - x_j|))}{\sum_{j=1}^{n} A_{ij}\exp(\mathrm{ReLU}(a^T|x_i - x_j|))}$$

Summary Questions of the lecture

- Describe the key aspects of Graph Diffusion-Embedding Networks.
- Describe the key aspects of Graph Diffusion Convolution.
- Describe the key aspects of Graph Learning-Convolutional Networks .