Brief Survey: Graph Neural Network Methods

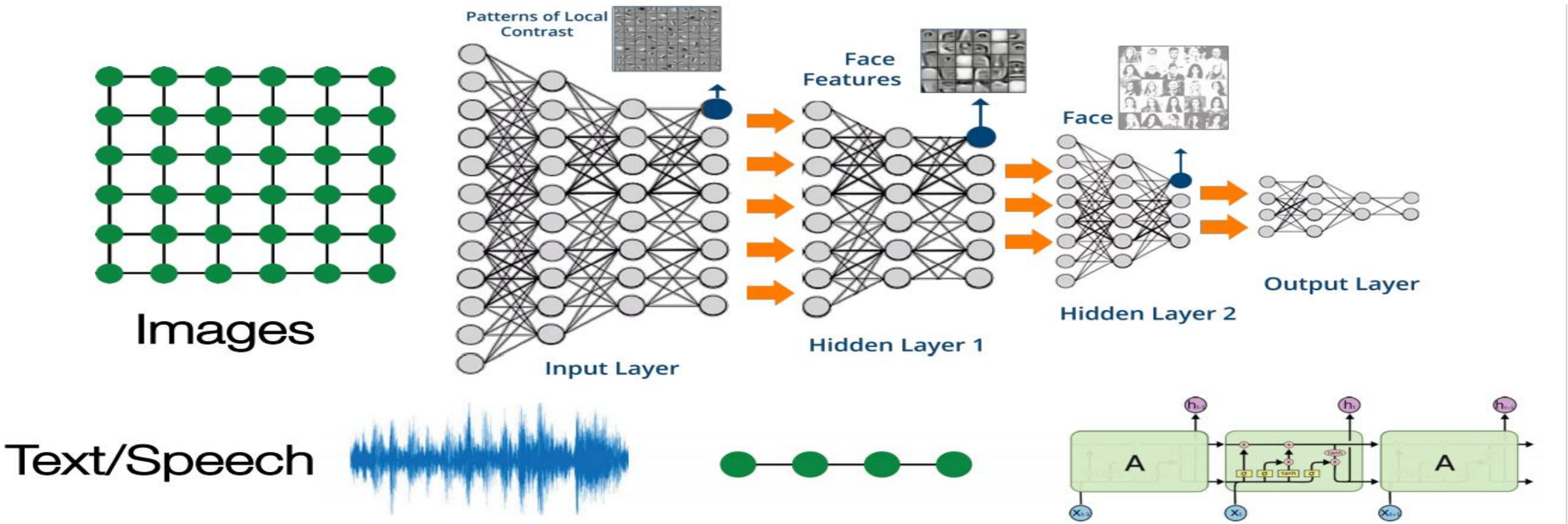
Gao Zhen

2021/03/19

Content

- Why GNN
- Classical Architectures of GNN
- An example of GNN in Recommendation System
- Future Work

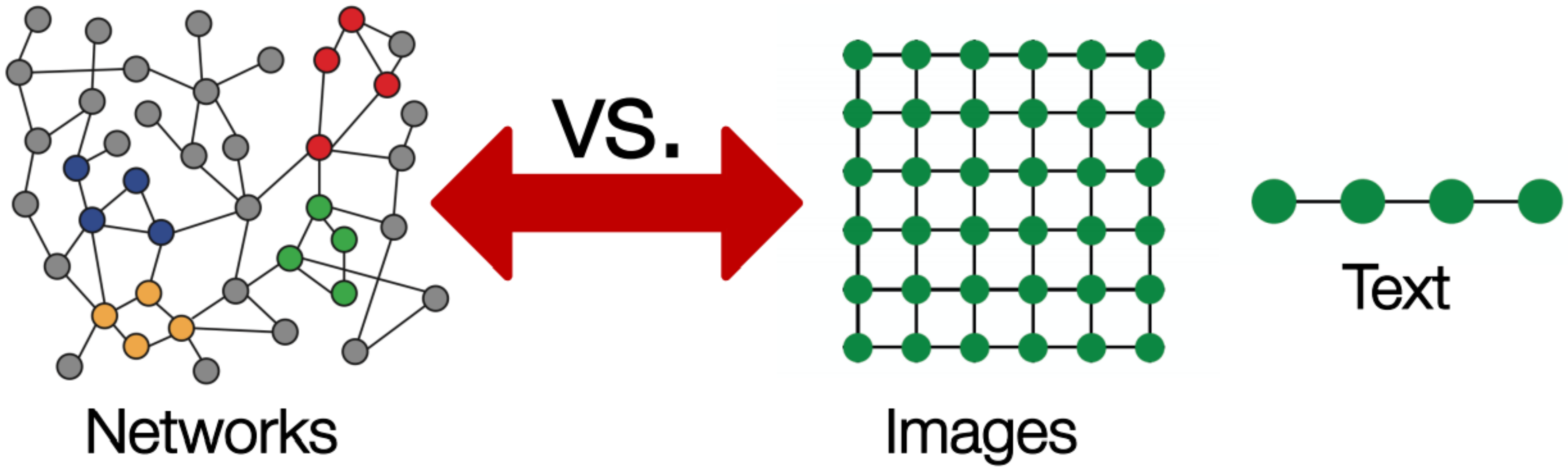
Modern ML Toolbox



Modern deep learning toolbox is designed for simple sequences & grids

Networks are Complex

- Arbitrary size and complex topological structure



- No fixed node ordering or reference point
- Often dynamic and have multimodal features

Tasks on Networks

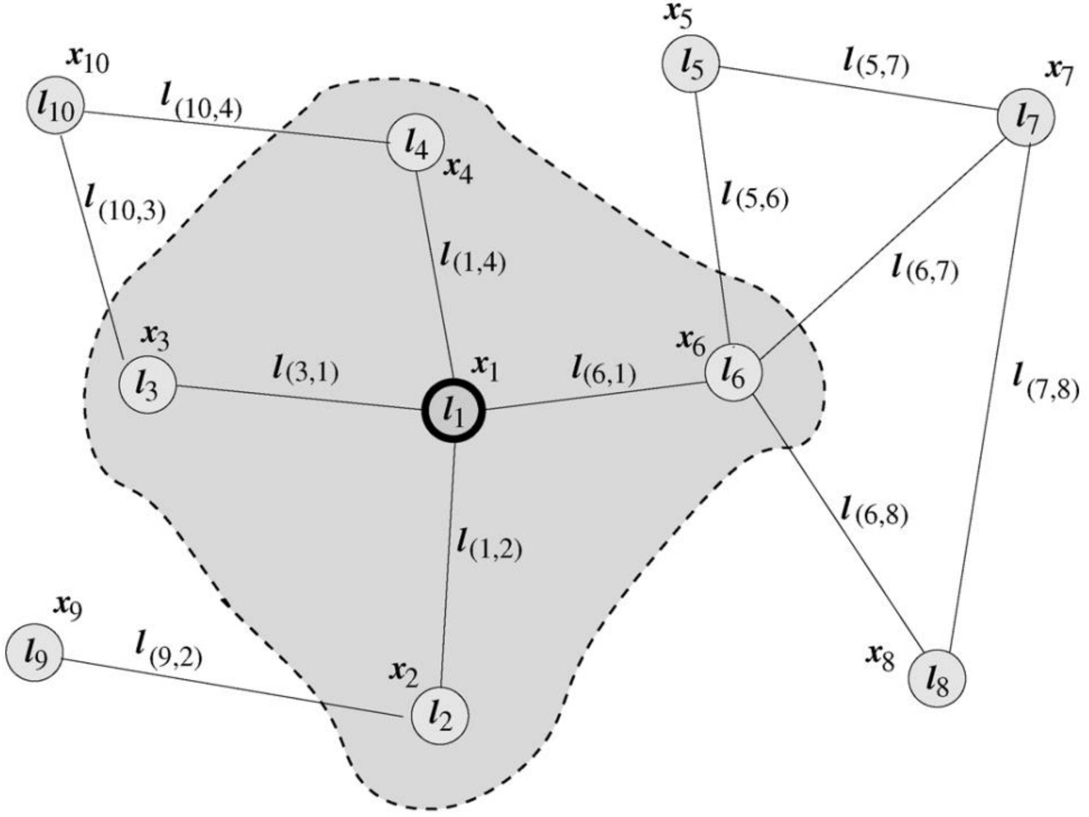
- Node classification
  - Predict a type of a given nod
- Link prediction
  - Predict whether two nodes are linked
- Community detection
  - Identify densely linked clusters of node
- Network similarity
  - How similar are two (sub)network

First GNN

$$\boldsymbol{x}_n = f_{\boldsymbol{w}}(\boldsymbol{l}_n, \boldsymbol{l}_{\mathrm{co}[n]}, \boldsymbol{x}_{\mathrm{ne}[n]}, \boldsymbol{l}_{\mathrm{ne}[n]})$$
$$\boldsymbol{o}_n = g_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{l}_n)$$

$$\boldsymbol{x} = F_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{l})$$
$$\boldsymbol{o} = G_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{l}_{\boldsymbol{N}})$$

- Like Recurrent Neural Network
- Banach fixed point theorem
  - F**w**(·) must be a “contraction map”
  - ∃μ, 0 < μ < 1
    ||F**w**(**x**, **l**) - F**w**(**x**, **l**)|| ⩽ μ||**x** - **y**||

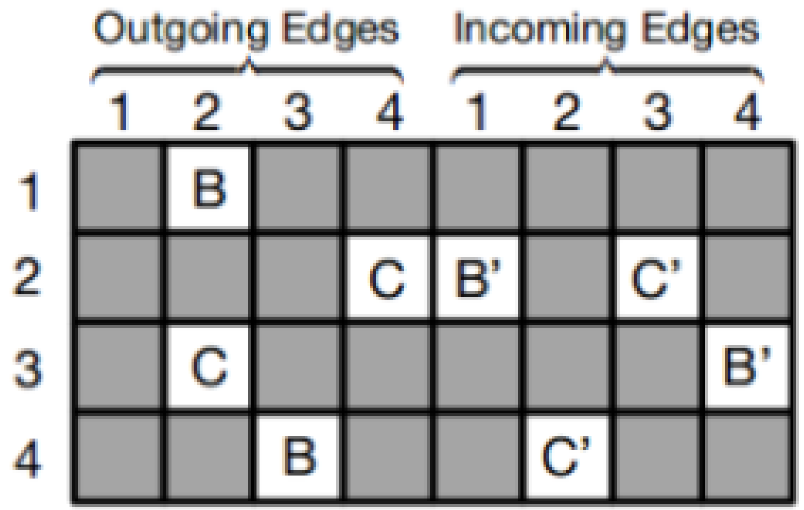


$$x_1 = f_{\mathrm{w}}(l_1, \underbrace{l_{(1,2)}, l_{(3,1)}, l_{(1,4)}, l_{(6,1)}}_{l\ co[1]}, \underbrace{x_2, x_3, x_4, x_6}_{x\ ne[1]}, \underbrace{l_2, l_3, l_4, l_6}_{l\ ne[n]})$$

[1]The graph neural network model. IEEE 2009

GGNN

- Based [1]
- Gated recurrent units
    - Cancle “contraction map”



$$\mathbf{A} = \left[\mathbf{A}^{(\text{out})}, \mathbf{A}^{(\text{in})}\right]$$

$$\mathbf{h}_v^{(1)} = [\boldsymbol{x}_v^\top, \mathbf{0}]^\top \quad (1)$$

$$\mathbf{a}_v^{(t)} = \mathbf{A}_{v:}^\top \left[\mathbf{h}_1^{(t-1)\top} \dots \mathbf{h}_{|\mathcal{V}|}^{(t-1)\top}\right]^\top + \mathbf{b} \quad (2)$$

$$\mathbf{z}_v^t = \sigma\left(\mathbf{W}^z \mathbf{a}_v^{(t)} + \mathbf{U}^z \mathbf{h}_v^{(t-1)}\right) \quad (3)$$

$$\mathbf{r}_v^t = \sigma\left(\mathbf{W}^r \mathbf{a}_v^{(t)} + \mathbf{U}^r \mathbf{h}_v^{(t-1)}\right) \quad (4)$$

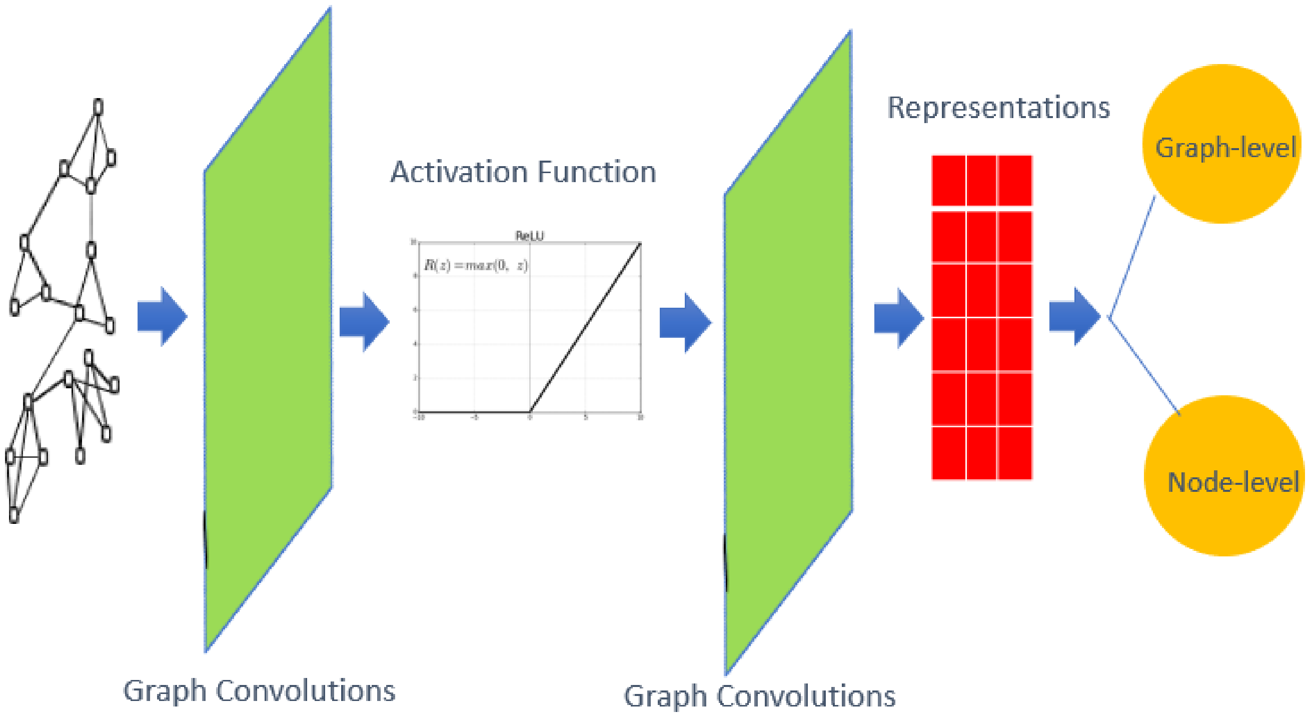
$$\widetilde{\mathbf{h}_v^{(t)}} = \tanh\left(\mathbf{W}\mathbf{a}_v^{(t)} + \mathbf{U}\left(\mathbf{r}_v^t \odot \mathbf{h}_v^{(t-1)}\right)\right) \quad (5)$$

$$\mathbf{h}_v^{(t)} = (1 - \mathbf{z}_v^t) \odot \mathbf{h}_v^{(t-1)} + \mathbf{z}_v^t \odot \widetilde{\mathbf{h}_v^{(t)}}. \quad (6)$$

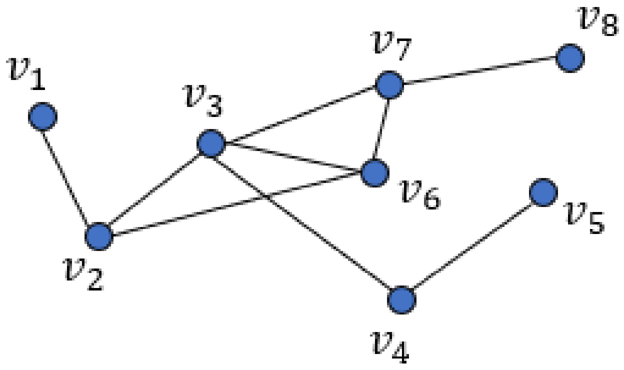
[2] Gated graph sequence neural networks. ICLR 2016

Graph Convolution Neural Networks

- Spectral Convolution
  - Laplacian matrix
  - Graph Fourier Transform
  - ChebNet
  - …
- Spatial Convolution
  - GraphSAGE
  - GAT
  - …



Matrix Representations of Graphs



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

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

Degree Matrix | Adjacency Matrix | Laplacian Matrix

$$\underset{\boldsymbol{D}}{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}} - \underset{\boldsymbol{A}}{\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}} = \underset{\boldsymbol{L}}{\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}}$$

Eigen-decomposition of Laplacian Matrix

- Laplacian matrix has a complete set of orthonormal eigenvectors：

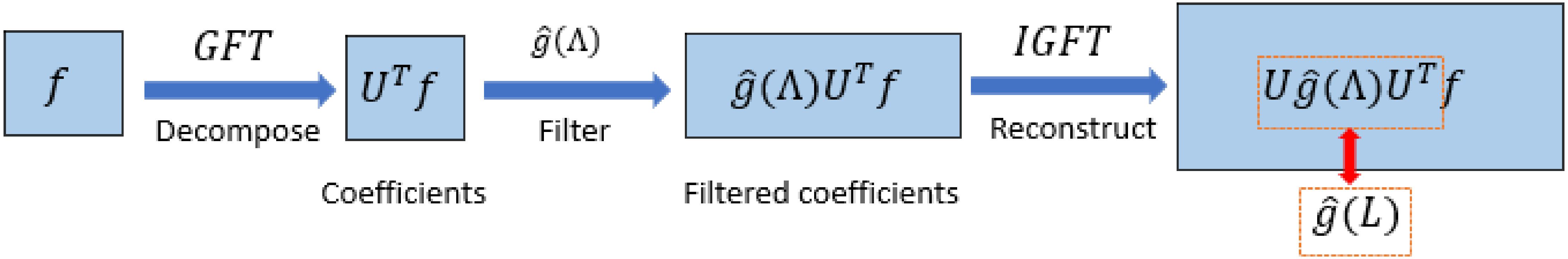
$$\mathbf{L} = \underbrace{\begin{bmatrix} | & & | \\ \mathbf{u}_0 & \cdots & \mathbf{u}_{N-1} \\ | & & | \end{bmatrix}}_{\boldsymbol{U}} \underbrace{\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix}}_{\boldsymbol{\Lambda}} \underbrace{\begin{bmatrix} — & \mathbf{u}_0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

- Eigenvalues are sorted non-decreasingly：

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

Spectral Convolution

- Graph Fourier Transform (GFT)
  - From spatial domain to spectral domain: $\hat{\boldsymbol{f}} = \boldsymbol{U}^T \boldsymbol{f}$
- Inverse Graph Fourier Transform (IGFT)
  - From spectral domain to spatial domain: $\boldsymbol{f} = \boldsymbol{U}\hat{\boldsymbol{f}}$



Graph Spectral Filtering for GNN

$$f \xrightarrow{\text{Filtering}} \hat{g}(L)f$$

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

- Example:

$$\hat{g}(\Lambda) = \begin{bmatrix} \theta_1 & & & \\ & \theta_2 & & \\ & & \dots & \\ & & & \theta_N \end{bmatrix}$$

$$\hat{g}(\Lambda) = \begin{bmatrix} \sum_{k=0}^{K} \theta_k \lambda_1^k & & & \\ & \sum_{k=0}^{K} \theta_k \lambda_2^k & & \\ & & \dots & \\ & & & \sum_{k=0}^{K} \theta_k \lambda_N^k \end{bmatrix}$$

$$U\hat{g}(\Lambda)U^T f$$

$$U\hat{g}(\Lambda)U^T f(i) = \sum_{j=0}^{N} \sum_{k=0}^{K} \theta_k L_{i,j}^k f(j)$$

ChebNet

- Chebyshev polynomials
  - Recursive definition： $T_0(x) = 1; T_1(x) = x$
    $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$

$$g(x) = \theta_0 T_0(x) + \theta_1 T_1(x) + \theta_2 T_2(x) + \cdots$$

- No eigen-decomposition needed
- Stable under perturbation of coefficients

$$\hat{g}(\Lambda) = \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda}), with\ \tilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - 1$$

[3]Convolutional neural networks on graphs with fast localized spectral filtering. NIPS 2016

GCN: Simplified ChebNet

- Use Chebyshev polynomials with K=1 and assume λ_max=2

$$\hat{g}(\Lambda) = \theta_0 + \theta_1(\Lambda - I)$$

- Further constrain $\theta = \theta_0 = -\theta_1$

$$\hat{g}(\Lambda) = \theta(2I - \Lambda)$$

$$U\hat{g}(\Lambda)U^T f = \theta(2I - L)f = \theta\left(I + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\right)f$$

- Apply a renormalization trick

$$U\hat{g}(\Lambda)U^T f = \theta\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}\right)f, with\ \hat{A} = A + I$$

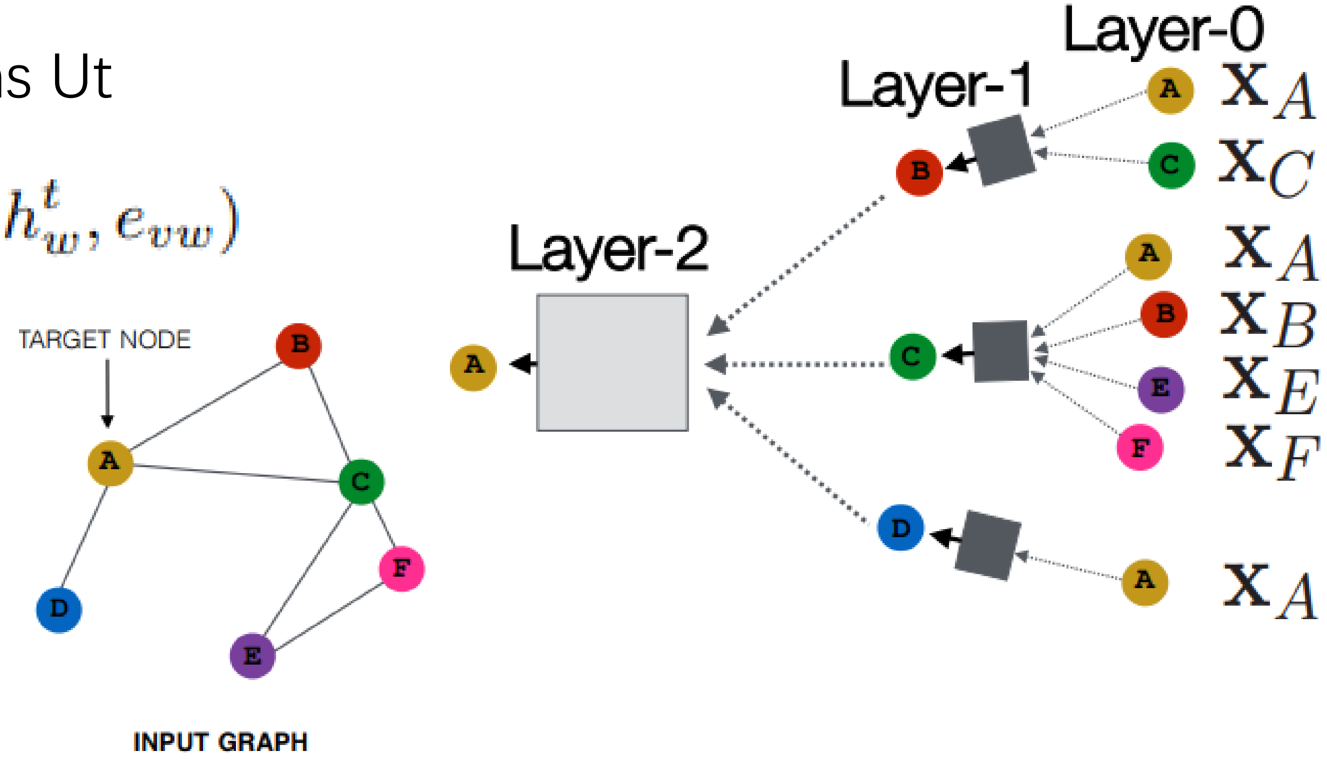
[4]Semi-supervised classification with graph convolutional networks. ICLR 2017

Spatial Convolution

- Message Passing Neural Network
  - message function Mt
  - vertex update functions Ut

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$



[5]Neural message passing for quantum chemistry. ICML 2017

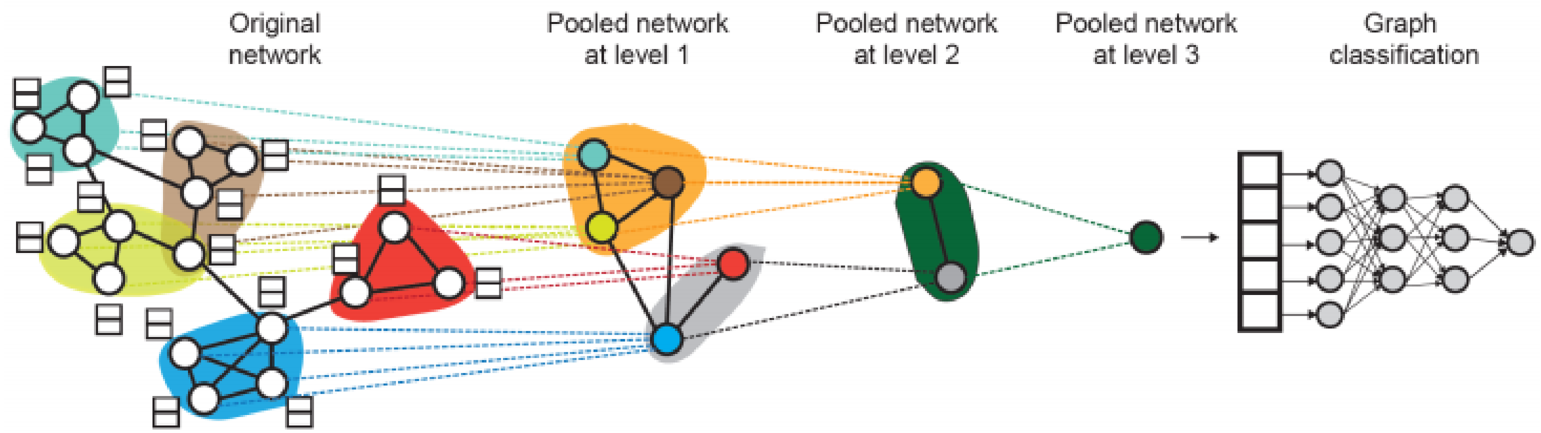
GraphSAGE

$$\mathbf{h}^k_{\mathcal{N}(v)} \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}^{k-1}_u, \forall u \in \mathcal{N}(v)\}); \qquad \mathbf{h}^k_v \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}^{k-1}_v, \mathbf{h}^k_{\mathcal{N}(v)})\right)$$

[6] Inductive representation learning on large graphs. NIPS 2017

DiffPool

- Hierarchically pool node embeddings



- Leverage 2 independent GNNs at each level
    - GNN A: Compute node embeddings
    - GNN B: Compute the cluster that a node belongs to
- A and B at each level can be executed in parallel

[7] Hierarchical Graph Representation Learning with Differentiable Pooling. NIPS 2018

GAT

- The importantance of i's message to node j:

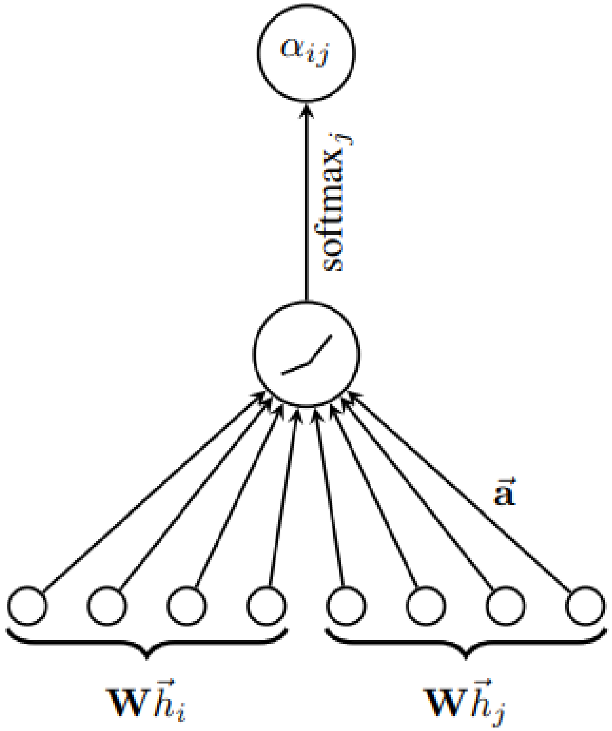
$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$

- Normalize into the final attention weight:

$$\alpha_{ij} = \mathrm{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}.$$

- Weight sum based on the final attention weight

$$\alpha_{ij} = \frac{\exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$



[8] Graph attention networks. ICLR 2018

JK-Nets

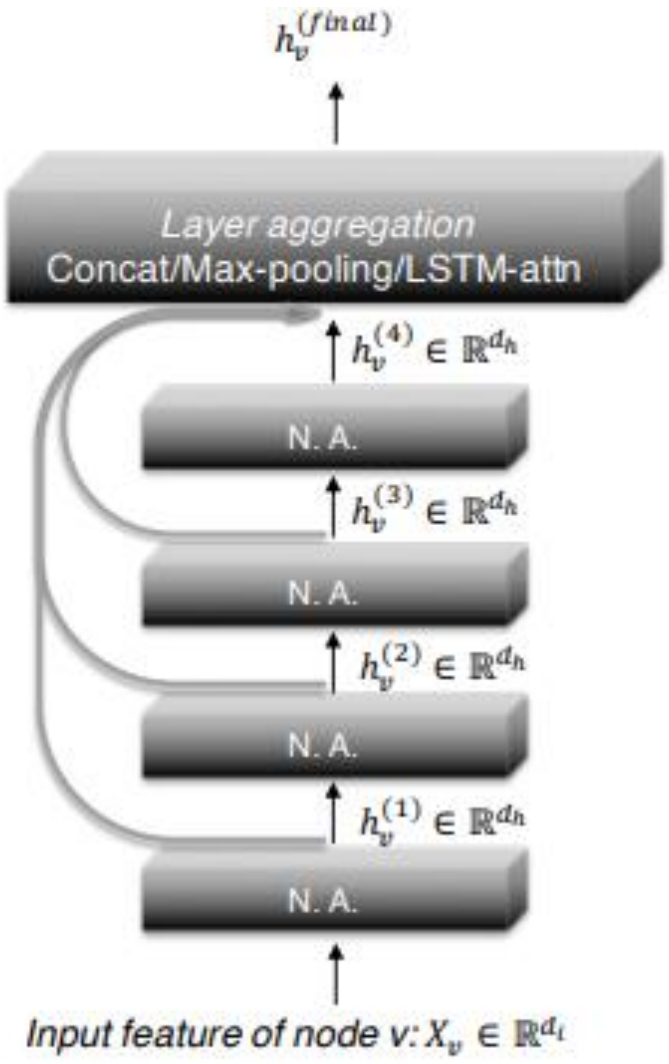
- Concatenation

$$\left[h_v^{(1)}, \ldots, h_v^{(k)}\right]$$

- Max-pooling
  - Element-wise

$$\max\left(h_v^{(1)}, \ldots, h_v^{(k)}\right)$$

- LSTM-attention
  - Put an attention score for each layer



[9]Representation learning on graphs with jumping knowledge networks. ICLR 2018

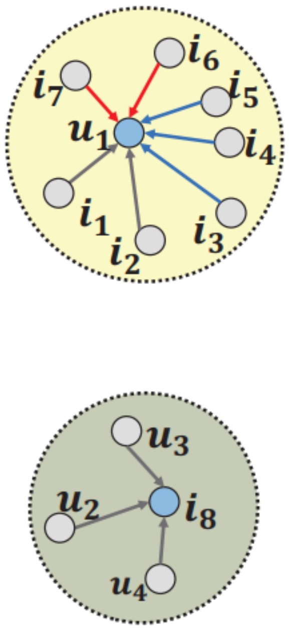
MBGCN

- User Embedding Propagation

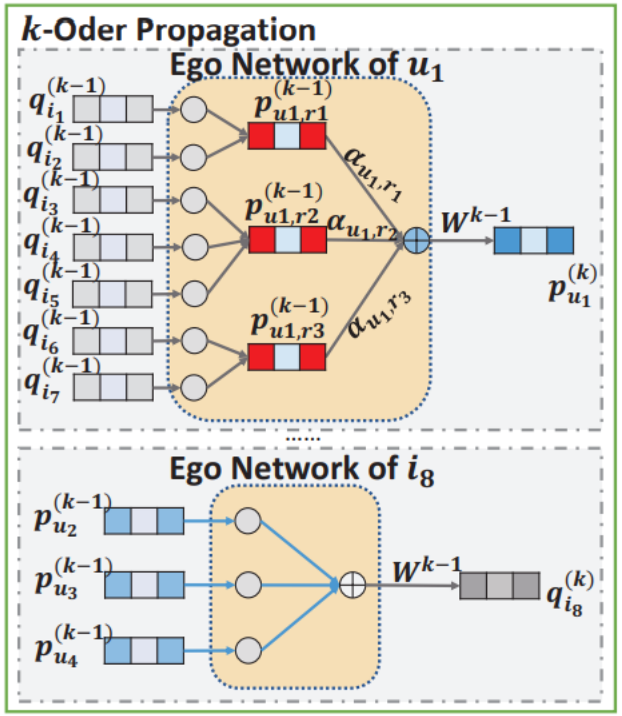
$$\alpha_{ut} = \frac{w_t \cdot n_{ut}}{\sum_{m \in N_r} w_m \cdot n_{um}}$$

$$p_{u,t}^{(l)} = \text{aggregate}(q_i^{(l)} | i \in N_t^I(u))$$

$$p_u^{(l+1)} = W^{(l)} \cdot (\sum_{t \in N_r} \alpha_{ut} p_{u,t}^{(l)})$$



(a) Local Graph



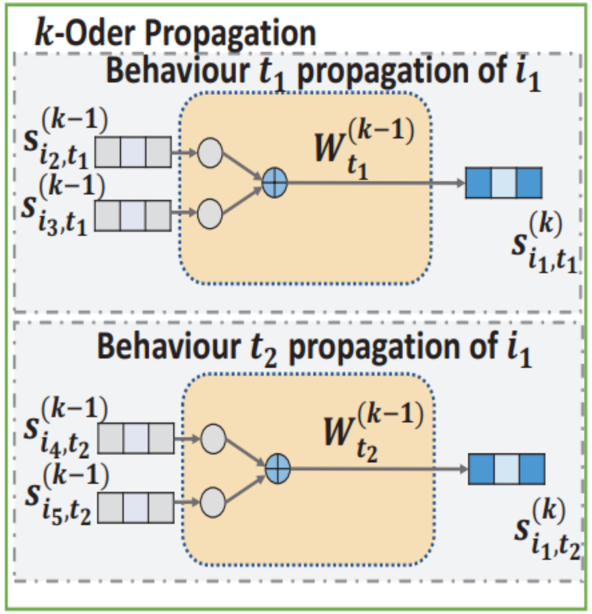
(b) k-order Propagation Process

- User-item layer learns the impact(weight) of different behaviors

MBGCN

- Item Embedding Propagation

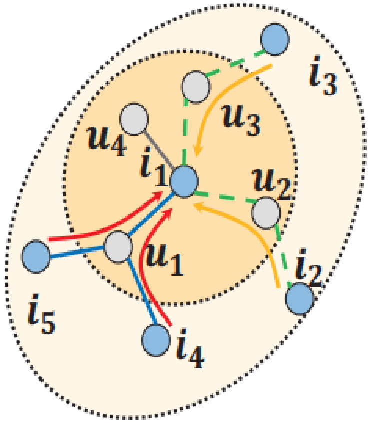
$$q_i^{(l+1)} = W^{(l)} \cdot \mathrm{aggregate}(p_j^{(l)} | j \in N^U(i))$$



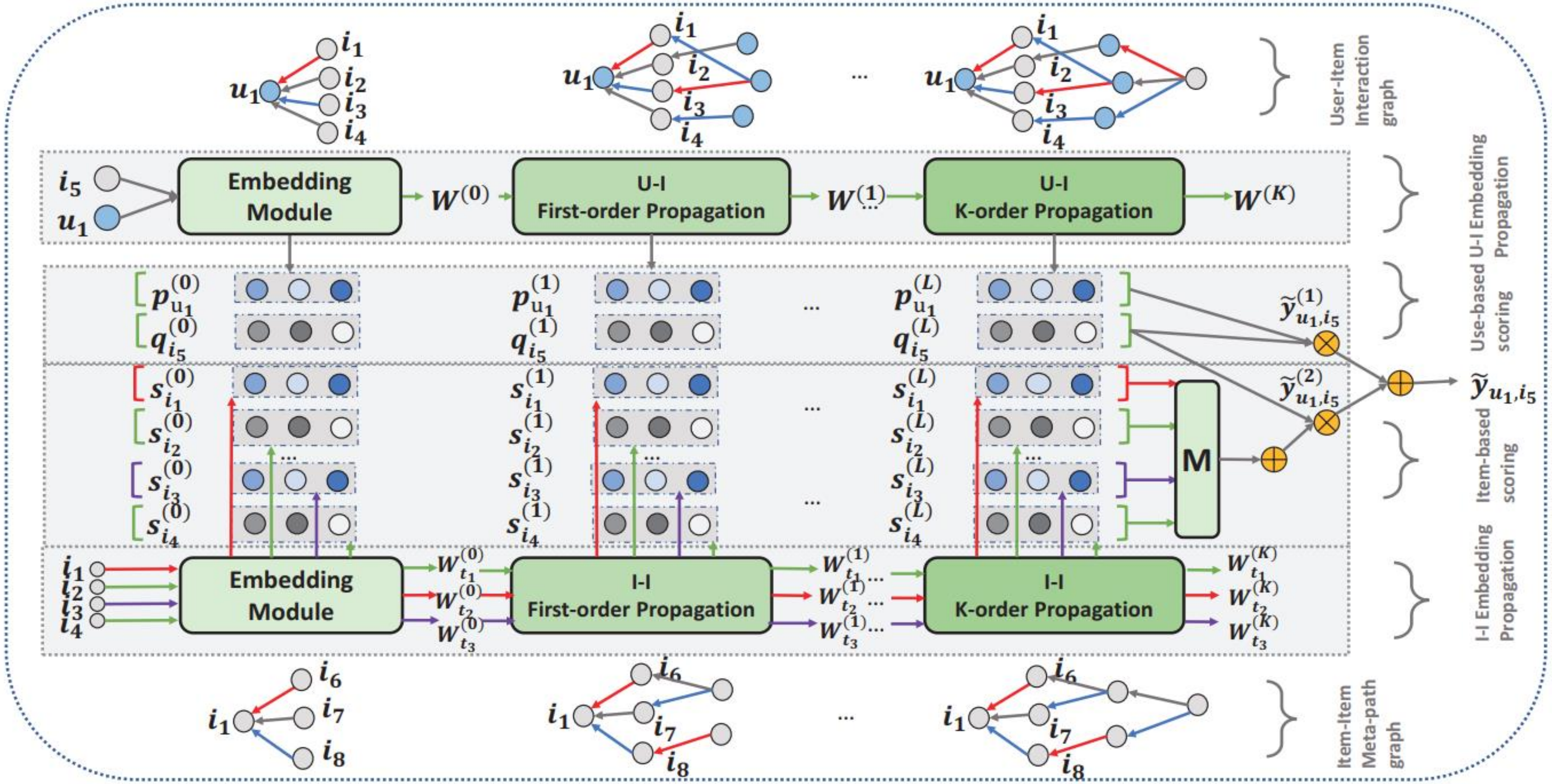
- Item-Relevance Aware Item-Item Propagation

$$s_{it}^{(l+1)} = W_t^{(l)} \cdot \mathrm{aggregate}(s_{jt}^{(l)} | j \in N_t^I(i))$$

- Item-item layer captures the semantics between behaviors



MBGCN



[10]Multi-behavior Recommendation with Graph Convolutional Networks. SIGIR 2020

Future Work

- About Semantic analysis of tables
  - The number of table cells' neighbors is certain
  - Key components
    - Filter Design
- May be Table Convolutional Neural Network(TNN)?
- About Recommendation System
  - Challenge:
    - Keep up with the research focus

Reference

[1]The graph neural network model. IEEE 2009.

[2]Gated graph sequence neural networks. ICLR 2016

[3]Convolutional neural networks on graphs with fast localized spectral filtering. NIPS 2016

[4]Semi-supervised classification with graph convolutional networks. ICLR 2017

[5]Neural message passing for quantum chemistry. ICML 2017

[6]Inductive representation learning on large graphs. NIPS 2017

[7]Hierarchical Graph Representation Learning with Differentiable Pooling. NIPS 2018

[8]Graph attention networks. ICLR 2018

[9]Representation learning on graphs with jumping knowledge networks. ICLR 2018

[10]Multi-behavior Recommendation with Graph Convolutional Networks. SIGIR 2020

[11]Design Space of Graph Neural Networks. NIPS 2020

[12]Identity-aware Graph Neural Networks. AAAI 2021

Thanks