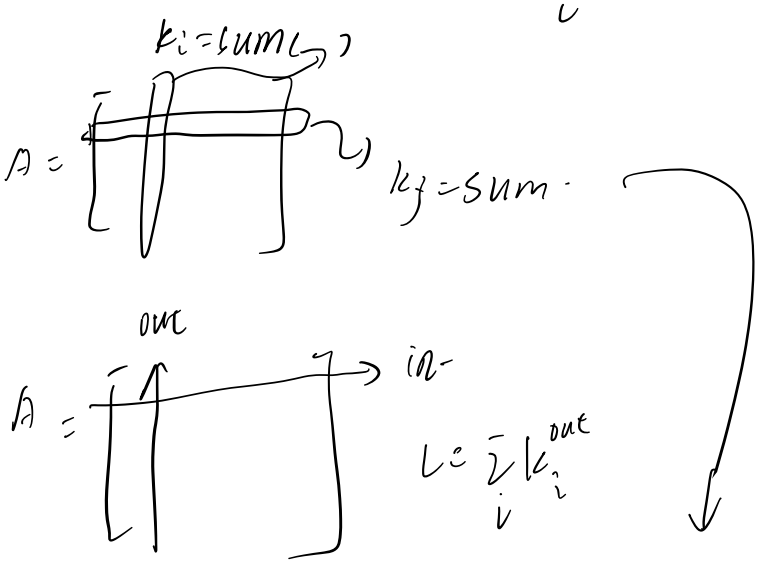
11. Intro.

Adj Matrix:

$L = \frac{1}{2}\sum_i k_i$

- undirect:



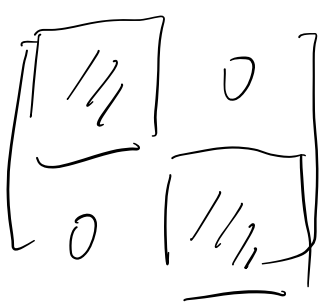
- direct:

平均度: $\bar{k} = \frac{2E}{N}$

weighted: $A = \begin{bmatrix} 0.1 & \cdots \\ 0.12 & \\ 1 & \end{bmatrix}$

$E = \frac{1}{2}\cdot\sum_{ij} B_{ij}$ $\quad$ $\bar{k} = 2\cdot\frac{E}{N}$

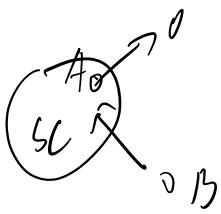
Disconnected:



Connectivity:

- Strongly connected: 任意2点间有通路 (A→B, B→A)
- Weakly ~ : ~ (不考虑方向)

SCCs:



A: In-component.

B: Out-component

L2, Feature learning.

- ML need features.

- Goal: learn Structure. position of a node.

Traditional Methods: { 1. Degree; 2, centrality; 3. Graphlets }

Node level:

- Degree: cons: can't capture importance of nodes.

- Centrality:

  - Degree: $C_d(v_i) = d_i$

$$\begin{cases} C_d^{norm}(v_i) = \frac{d_i}{n-1} \\ C_d^{max}(v_i) = \frac{d_i}{\max_j(d_j)} \\ C_d^{sum}(v_i) \approx \frac{d_i}{2|E|} \end{cases}$$

  - Betweenness:

$$C_B(v_i) = \sum_{j<k} \frac{g_{jk}(i)}{g_{jk}}$$

$g_{jk}(i)$ → paths j-k pass i

g_{jk} → all paths in j-k

$$C_B'(v_i) = \frac{C_B(v_i)}{\frac{(n-1)(n-2)}{2}}$$

- Closeness :

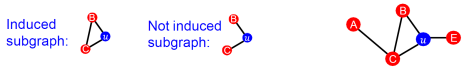
$$C_C(v_i) = \frac{N-1}{\sum_{j=1}^{N} d(i,j)}$$

Comparison:

	Low Degree	Low Closeness	Low Betweenness
High Degree		*Node is embedded in a community that is far from the rest of the network*	*Node's connections are redundant - communication bypasses the node*
High Closeness	*Key node connected to important/active alters*		*Probably multiple paths in the network, node is near many people, but so are many others*
High Betweenness	*Node's few ties are crucial for network flow*	*Very rare! Node monopolizes the ties from a small number of people to many others.*	

-Graphlets:

- Def: Induced subgraph is another graph, formed from a subset of vertices and *all* of the edges connecting the vertices in that subset.



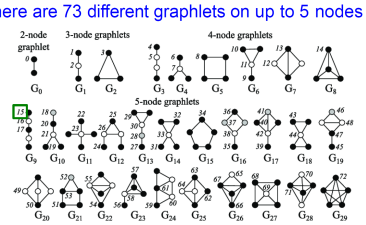
Homogeneous			Heterogenous					
Pattern	Graph	Count	Pattern	Graph	Count	Pattern	Graph	Count
		0			0			0
		12			4			1
		24			6			2

- Graphlets: Rooted connected induced non-isomorphic subgraphs:

There are 73 different graphlets on up to 5 nodes

Take some nodes and all the edges between them.

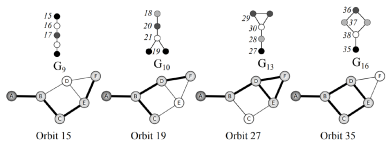
Graphlet id (Root / "position" of node *u*)



Note: Here is still on homogeneous graphs. Different colours distinguish different orbits and positions.

Graphlet Degree Vector

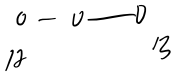
- Computation of the graphlet degree vector (GDV) of node A in the friendship network
  - GDV provides a measureof a node's local network topology
  - Comparing vectors of two nodes provides a more detailed measure of local topological similarity than node degrees



Orbit	0	1	2...3	4	5	6	7...14	15	16...18	19	20...26	27	28...34	35	36...72
GDV(A)	1	2	0...0	3	0	1	0...0	1	0...0	1	0...0	1	0...0	1	0...0

Link level: 預測 edge score : $C(x,y)$

1. Distance:



$S_{12,13} = 2$.

2. Local Neighbourhood Overlap:

$$\text{Jaccard} = \frac{N(12) \cap N(13)}{N(12) \cup N(13)}$$

L3 PageRank.

- $r = Mr$

- 计算：令 $r^0 = [1/n, \cdots 1/n]^T$

$$r^{k+1} = Mr^k$$

直至：$\|r^{k+1} - r^k\|_1 < \varepsilon$.

问题1：spider trap

→ ③ ②，会导致 PR 最高、其他为 0

Solution：

$$r^{k+1} = \beta \cdot M \cdot r^k + (1-\beta)\begin{bmatrix} 1/n \\ \vdots \\ 1/n \end{bmatrix}$$

（网页会随机跳到其它网页.

问题2：Dead End

→ ③，会导致 rank 全是 0. (Non-stochastic!)

Solution: $\begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1/n \\ \vdots \\ 1/n \end{bmatrix}$

计算问题：M 非常 sparce. M 的空间太大. $M \approx 10N$ entries.

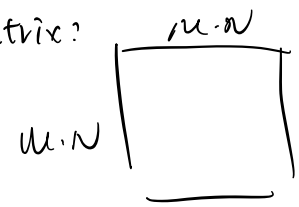
Solution：利用 adj list，encode M.

公式：$r \leftarrow \beta Mr + (1-\beta)[1/N]_N$

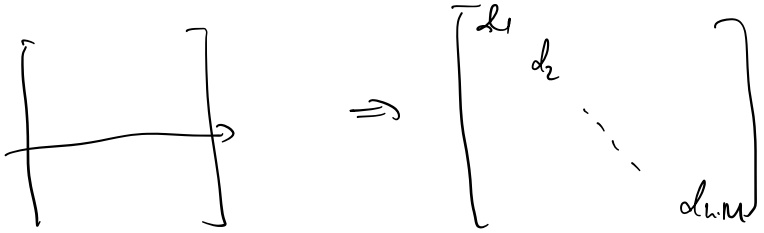
14. Graph Laplacian.

- similarity: $W_{ij} = e^{-\frac{\|x_{(i)} - x_{(j)}\|_2^2}{\sigma^2}}$

- Affinity Matrix:

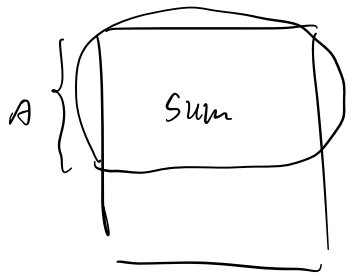


- Degree:



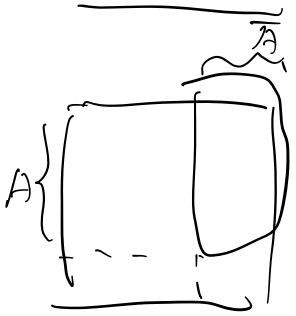
- Volume of set:

$$Vol(A) = \sum_{i \in A} d_i$$



- Cuts:

$$Cut(A, \bar{A}) = \sum_{i \in A, j \in \bar{A}} W_{ij}$$



- Laplace: $L = D - W$

- Spetral Clustering:

- Min cut:

$$MinCut(A, B) = \min_{A,B} \sum_{u \in A, v \in B} w(u, v)$$

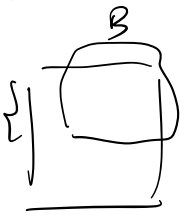
(intuition：两团体 i 间连接尽可能松)

—Normalized Cut:

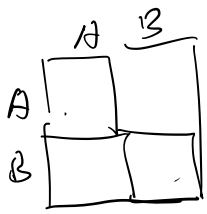
$$NCut(A,B) = Cut(A,B)\left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)}\right)$$

排除小的 outlier 干扰

association: $assoc(A,B) = \sum_{u\in A, v\in B} w(u,v)$



$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}$$



$$\Rightarrow Ncut(A,B) = 2 - Nassoc(A,B)$$

Problem: 2-clustering.

Goal: Find A,B. $\min_{A,B} Ncut(A,B)$

NP-hard

Solution: $\min_x Ncut(x) = \min_y \frac{y^T(D-W)y}{y^TDy}$, Subject to $y^TD\vec{1} = 0$

$\Leftrightarrow \min_y y^T(D-W)y$ subject to $y^TDy = 1$

$\Leftarrow \frac{\partial\, y^T(D-W)y - \lambda(y^TDy-1)}{\partial y} = 0$

$\Leftrightarrow (D-W)y - \lambda Dy = 0$

$(D-W)y = \lambda Dy$

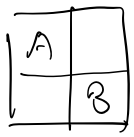
$\Rightarrow D^{-1}(D-W)y = \lambda y$

$(I - D^{-1}W)y = \lambda y$ $\Rightarrow$ eigenvector.

(second smallest)

$\sum_i y_i D_{ii} = 0$

- Avg association: $\max_{A,B} \left\{ \frac{assoc(A,A)}{|A|} + \frac{assoc(B,B)}{|B|} \right\}$



- Avg cut: $\min_{A,B} \left\{ \frac{Cut(A,B)}{|A|} + \frac{Cut(A,B)}{|B|} \right\}$

- Laplacian: $L = D - W$.

$$L_{sym} = D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}$$

$$L_{rw} = D^{-1}(D-W)$$

$$L_{sym} = D^{\frac{1}{2}} L_{rw} D^{-\frac{1}{2}}$$

- $y^T(D-W)y = \sum_{i,j} w_{ij}(y_i - y_j)^2$ — smoothness of G

$$y^T D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}} y = \sum_{i,j} w_{ij}\left(\frac{y_i}{\sqrt{D_{ii}}} - \frac{y_j}{\sqrt{D_{jj}}}\right)^2$$

- $(Ly)_i = \sum_{j \in N(i)} w_{ij}(y_i - y_j)$

- L^{-1} as operator:
  - $(D^{-1}W)^k$ k-th hop
  - $\sum_{k=0}^{\infty} (D^{-1}W)^k = I - D^{-1}W$

计算过程

$(Ay)_i = \sum_j A_{ij} y_j$

$\Rightarrow y^T Ay = \left(\sum_{ij} A_{ij} y_j\right) \cdot y_i$

$= \sum_{ij} A_{ij} y_j \cdot y_i$

$y^T Ay = \sum_i \sum_j A_{ij} y_i y_j$

$\frac{\partial y^T Ay}{\partial y_k} = \frac{\partial}{\partial y_k} \sum_i \sum_j A_{ij} y_i y_j$

$= \sum_{j \neq k} A_{kj} y_j + 2A_{kk} y_k + \sum_{i \neq k} A_{ik} y_i$

$= \sum_j A_{kj} y_j + \sum_i A_{ik} y_i$

$\therefore \frac{\partial y^T Ay}{\partial y} = y^T D + y^T D^T = y^T (D + D^T)$

$\frac{\partial y^T (D-W) y}{\partial y} = y^T \left[(D-W) + (D-W)^T\right]$

$= y^T (D - W + D^T - W^T)$

$= 2 y^T (D - W)$

$\Rightarrow (D-W) y - \lambda D y = 0$

$y^T (D-W) y$

$\left((D-W)y\right)_i = \sum_j (D-W)_{ij} y_j$

$\Rightarrow y^T (D-W) y = \left(\sum_j (D-W)_{ij} y_j\right) y_i$

$= \sum_{ij} D_{ij} y_i y_j - \sum_{ij} W_{ij} y_i y_j$

$= \sum_i D_{ii} y_i^2 - \sum_{ij} W_{ij} y_i y_j$

$= \sum_i \sum_j W_{ij} y_i^2 - \sum_{ij} W_{ij} y_i y_j$

$= \frac{1}{2} \sum_{ij} W_{ij} y_i^2 - \sum_{ij} W_{ij} y_i y_j + \frac{1}{2} \sum_{ij} W_{ij}$

$= \frac{1}{2} \sum_{ij} W_{ij} (y_i - y_j)^2$

Cuts Comparison:

Finding clumps ⟷ Finding splits

	Average association	Normalized Cut	Average cut
Discrete formulation	$\frac{asso(A,A)}{\lvert A\rvert} + \frac{asso(B,B)}{\lvert B\rvert}$	$\frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V)}$ or $2 - \left(\frac{asso(A,A)}{asso(A,V)} + \frac{asso(B,B)}{asso(B,V)}\right)$	$\frac{cut(A,B)}{\lvert A\rvert} + \frac{cut(A,B)}{\lvert B\rvert}$
Continuous solution	$Wx = \bar{\lambda} x$	$(D-W)x = \bar{\lambda} D x$ or $W x = (1 - \bar{\lambda}) D x$	$(D-W)x = \bar{\lambda} x$

$$
\begin{aligned}
& f^T D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}} f \\
&= f^T f - f^T D^{-\frac{1}{2}} W D^{-\frac{1}{2}} f \\
&= \sum_{i=1}^{N} f_i^2 - \begin{bmatrix} \vdots \\ \frac{f_i}{\sqrt{D_{ii}}} \\ \vdots \end{bmatrix} \cdot \begin{bmatrix} \cdots & \sum_{j=1}^{N} \frac{f_j}{\sqrt{D_{jj}}} \cdot w_{ij} & \cdots \end{bmatrix} \rightarrow \text{i-th row} \\
&= \sum_{i=1}^{N} f_i^2 - \sum_{i=1}^{N} \frac{f_i}{\sqrt{D_{ii}}} \cdot \sum_{j=1}^{N} \frac{f_j}{\sqrt{D_{jj}}} w_{ij} \\
&= \sum_{i=1}^{N} \sum_{j=1}^{N} f_i^2 - \frac{f_i \cdot f_j}{\sqrt{D_{ii}} \cdot \sqrt{D_{jj}}} \cdot w_{ij} \\
&= \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} \left(\frac{f_i^2}{D_{ii}} - \frac{2 f_i \cdot f_j}{\sqrt{D_{ii}} \sqrt{D_{jj}}} + \frac{f_j^2}{D_{jj}} \right) \\
&= \frac{1}{2} \sum_{i,j=1}^{N} w_{ij} \left(\frac{f_i}{\sqrt{D_{ii}}} - \frac{f_j}{\sqrt{D_{jj}}} \right)^2
\end{aligned}
$$

- Personalised PageRank:
  - $p^{t+1} = (1-\beta) q + \beta W p^t$ (不同的是，会有一定概率回到原点)
  - 用于 semi-supervised learning. (回到有 label 的地方)
  - eg: (2 classes)

    Formula: $f^{t+1} = (1-\alpha) y + \alpha S f^t$ ($S = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$)

    $\Rightarrow f^{t+1} = (1-\alpha) y + \alpha S((1-\alpha) y + \alpha S f^{t-1})$

$$= (1-\alpha)y \cdot \sum_{i=0}^{t} (\alpha S)^t + (\alpha S)^t y \quad (\to v)$$

$$\lim_{t\to\infty} \sum_{t} (\alpha S)^t = \frac{1-(\alpha S)^{t+1}}{1-\alpha S} = (1-\alpha S)^{-1}$$

$$\therefore f^* = \lim_{t\to\infty} f^t = (1-\alpha)\cdot y (1-\alpha S)^{-1}$$

In fact: $L(f) = \frac{1}{2}\left(\sum_{ij} w_{ij} \left| \frac{f_i}{\sqrt{D_{ii}}} - \frac{f_j}{\sqrt{D_{jj}}} \right|^2 + \mu \sum_i \| f_i - y_i \|^2 \right)$

$$= \frac{1}{2}\left(f^T D^{-\frac{1}{2}} (D-W) D^{-\frac{1}{2}} f + \mu \cdot (f-y)^T (f-y) \right)$$

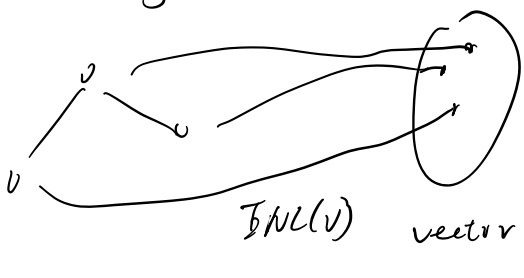
$$\frac{\partial L(f)}{\partial f} = 0 \iff \underbrace{D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}}_{A} f + \mu \cdot (f-y) = 0$$

$$\iff f^* = \mu y \cdot (D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}} + \mu \cdot I)^{-1}$$

(这是 GCN 之前 . semisupervised learning 的常规做法)

25. Node / Graph Embedding.

1. Node Embedding.



- $sim(z_1, z_2) \approx z_1^T z_2$
- learn structure feature, embedding for downstream task.
- 2种寻找nodes间关系的办法：

① Adj-based similarity:

$$-L = \sum_{(u,v) \in V \times V} \| z_u^T z_v - A_{u,v} \|^2$$

- 用SGD更新.
- Cons: 1. $O(|V|^2)$ time. (改进：由于是sparse图，对non-zero更新.)
  2. 只考虑local, direct连接

② Random Walk:

- why? 1. Expressive：考虑了local, high-order的信息.
  2. 只考虑Rand Walk上的node

· $N_R(u)$ — set of neighbur nodes. R — strategy

· objective: $\max_f \sum_{u \in V} \log P(N_R(u) | z_u)$

$$\Leftrightarrow L = -\sum_{u \in V} \sum_{v \in N(u)} \log P(v | z_u)$$

where $P(v|z_u) = \dfrac{\exp(z_v^T \cdot z_u)}{\sum_{k \in V} \exp(z_k^T \cdot z_u)}$

$$\Rightarrow L = -\sum_{u \in V} \sum_{v \in N(u)} \log \frac{\exp(z_v^T \cdot z_u)}{\sum_{k \in V} \exp(z_k^T \cdot z_u)}$$

$O(|V|^2)$!

· Word2Vec 2种 model: v_1, v_2, v_3, v_4, v_5

1. CBOW: 背景词 → 中心词

$v_2 \uparrow \xrightarrow{z_{V \times N}} z_2 \searrow$

$\hat{z} = \frac{z_2 + z_4}{2} \xrightarrow{z_{N \times V}} \hat{y} \sim y_3$

$v_4 \uparrow \xrightarrow{z_{V \times N}} z_4 \nearrow$

2. SG: 中心词 → 背景词

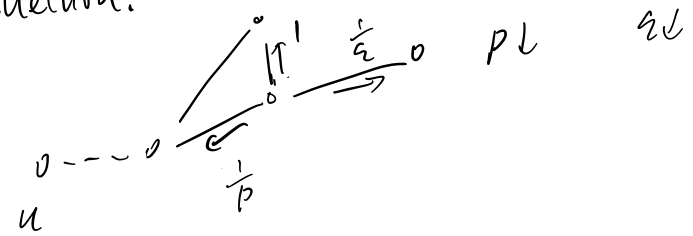
$v_3 \uparrow \xrightarrow{z_{V \times N}} z_3 \xrightarrow{z_{N \times V}} \hat{y} \sim y_2$ or y_4

- 如何解决 $O(|V|^2)$?
  - Approximation:

$$\log \frac{\exp(z_v^T \cdot z_u)}{\sum_{n \in V} (z_n^T \cdot z_u)} \approx \log \sigma(z_u^T z_v) - \sum_{k} \log \sigma(z_u^T z_k)$$

k 个 negative sample

  - k ↑, robust ↑
  - k nodes are not on the walks!
- Better strategy: Biased Walks.
  - why? trade off between local / global
  - Method:



$$\pi_{c_i|c_{i-1},c_{i-2}} = \begin{cases} 1, & d(c_{i-2}, c_i) = 1 \\ 1/p, & \sim = 2 \\ 1/q, & \sim = 0 \end{cases}$$

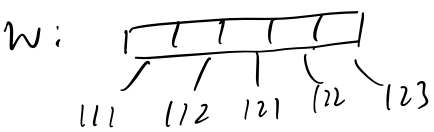
2. Graph Embedding:

法一：sum / avg of node embedding $z_G = \sum_{v \in G} z_v$

法二：virtual node, and run node embedding

法三：Anonymous Walk Embedding.

- 1. do m times random walk with len l
  2. record w_i counts
  3. normalize w vector. w is G embedding.



- How many walks?

$$m = \left\lceil \frac{2}{\varepsilon^2}\left(\log(2^\eta - 2) - \log(\delta)\right) \right\rceil$$

η: theoretically existed types of asym. walks of len l.

δ: $P(\#\text{errors} \le \varepsilon) < \delta$

2b GNN.

- Limitation of shallow Embedding:
  1. $O(|V|)$ params
  2. can't generate new embedding for new nodes.
  3. no data feature.
- Goal: Find $f: V \to R^d$

- Naive: MLP($[A_i, f_i]$)
  问题: 1. $O(|V|)$ parameters
  2. not invariant to node ordering
- GNN model should obey Permutation Equivariant/Invariant:
  1. Permutation Equivariant:
     同一幅图, 不同 order 的 (A, X), (A', X'), $f: G \to R^d$: vector
     对于同一个 node n: $f(A, X)_n = f(A, X')_n$
     $\Leftrightarrow$ f is permutation equivariant.
  2. Permutation Invariant:
     同一幅图, 不同 order 的 (A, X), (A', X'), $f: G \to k$: score.
     $f(A, X) = f(A', X')$
     $\Leftrightarrow$ f is Permutation Invariant.

· 2 Models:

1. Basic GNN:

· Math:

$$h_v^0 = x_v$$

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

· Matrix Formulation:

$$H^{(k+1)} = \sigma\left(\tilde{D^{-1}A}H^{(k)} W_k^T + H^k B_k^T\right)$$

$\tilde{D^{-1}A}$: aggregation. W_k^T: projection

· Train:

1. unsupervised: 利用图结构, node similarity
   · DeepWalk, Adj sim
2. supervised: $f: R^n \to R$. classification.

- Pros: 1. share paras
  2. inductive

2. GraphSage:

· $h_v^{(k)} = \sigma\left([W_k \cdot AGG(h_u^{k-1}, \forall u \in N(v)), B_k h_v^{k-1}]\right)$

- 变化：1. $AGG(\cdot)$ generalization

  2. sum → concatenation.

- 3 种 AGG：

  1. Mean

  2. Pool：$\gamma\left(\{Qh_u^{k-1}, \forall u \in N(v)\}\right)$ → element-wise mean / max

  3. LSTM：$LSTM\left([h_u^{k-1}, \forall u \in \pi(N(v))]\right)$

     π：生成一个随机序列

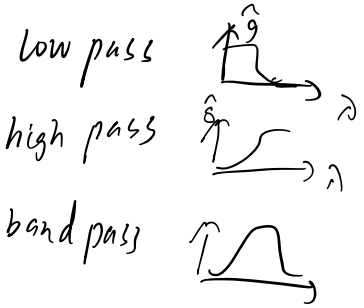
L7. GNN 2.

- Graph Signal: $f: V \to R^{N \times d}$
- Laplace Operator:
  1. $h = Lf$, $h(i) = \sum_{v_j \in N(v_i)} (f(i) - f(j))$
  2. $f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} A_{ij} (f_i - f_j)^2$
- $L = U \Lambda U^T$
  - $\Lambda = [\lambda_0, \ldots, \lambda_{N-1}]$, where $0 = \lambda_0 \le \lambda_1 \cdots \le \lambda_{N-1}$.
    - $L \cdot 1 = 0 \cdot 1$ (trivial solution)

$u_0 \ldots u_{n-1}$ are based signals, $\lambda_0 \ldots \lambda_{n-1}$ are corresponding frequency.

- Graph Fourier Transform: smoothness: $Lu = \sum_{ij} w_{ij} (u_i - u_j)$
  - $L = U \Lambda U^T$ ($Lu = \lambda u$) → frequency. → $|U| = 1$
  - GFT: $\hat{f} = U^T f$
  - Inverse GFT: $f = U \hat{f}$
- GNN Framework:
  - Filter: $A \in R^{n \times n}$, $X \in R^{n \times d} \Rightarrow A \in R^{n \times n}$, $X \in R^{n \times d_{new}}$
  - Pooling: $A \in R^{n \times n}$, $X \in R^{n \times d} \Rightarrow A \times R^{n_p \times n_p}$, $X \in R^{n_p \times d_{new}}$

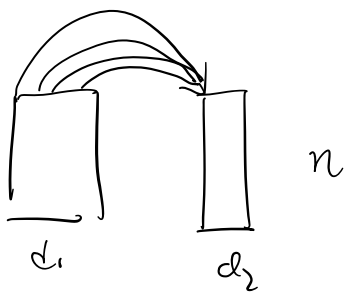
- Spectral Filtering:
  - $f \xrightarrow{GFT} U^T f \xrightarrow{\hat{g}(\Lambda)} g(\Lambda) U^T f \xrightarrow{IGFT} U \hat{g}(\Lambda) U^T f$
  - $\hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & & \\ & \ddots & \\ & & \hat{g}(\lambda_{n-1}) \end{bmatrix}$



  - $f^* \leftarrow U \underbrace{g(\Lambda) U^T f}_{\hat{g}(L)}$
  - we learn $g(\Lambda)$ from data!

- Multi-channel signals:

$F_{in} \in \mathbb{R}^{n \times d_1} \rightarrow F_{out} \in \mathbb{R}^{n \times d_2}$



$$F_{out}[:, i] = \sum_{j=1}^{d_1} \hat{g}_{i,j}(L) \cdot F_{in}[:, j], \quad i = 1, \ldots, d_2$$

∴ we have $d_2 \times d_1$ filters.

- $g(\Lambda) = \begin{bmatrix} \theta_1 & & \\ & \ddots & \\ & & \theta_n \end{bmatrix} \Rightarrow d_2 \times d_1 \times N$ parameters.

- Polynomial Parameterized:

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

$$L^K = (U \Lambda U^T)^K = U \Lambda^K U^T \rightarrow \begin{bmatrix} \lambda_1^K & & \\ & \ddots & \\ & & \lambda_N^K \end{bmatrix}$$

  - $d_2 \times d_1 \times K$ params.

  - $U \hat{g}(\Lambda) U^T f = \sum_{k=0}^{K} \theta_k L^k f$ (no eigen-decomposition needed)

  - Spatial View: $L^K f$: smoothness within k-hops.

- Chebyshev Polynomial:

  - $T_0(x) = 1$, $T_1(x) = x$, $T_{n+1}(x) = 2 \cdot T_n(x) - T_{n-1}(x)$

  - Moivre's formula: $(\cos x + i \cdot \sin x)^n = \cos nx + i \sin nx$

    (proved by induction.)

  - Othogonal polynomial (def)

    w/ integrable, $w(x) \geq 0$, $\langle f_m, f_n \rangle = \int_a^b f_m(x) \cdot f_n(x) w(x) \, dx$

$\langle f_m, f_n \rangle = 0$ for $m \neq n$.

- Gram-Schmidt orthonormalization:

$$\text{proj}_u(v) \overset{\text{def}}{=} \frac{\langle u, v \rangle}{\langle u, u \rangle} \cdot \vec{u}$$

$$u_1 = v_1$$
$$u_2 = v_2 - \text{proj}_{u_1} v_2$$
$$u_3 = v_3 - \text{proj}_{u_1} v_3 - \text{proj}_{u_2} v_3$$
$$\vdots$$
$$u_k = v_k - \sum_{i=1}^{k-1} \text{proj}_{u_i} v_k$$

then normalize $\{u_i\}_{i=1}^{k}$

- Def by trigonometric function:

$$T_n(\cos\theta) = \cos(n\theta)$$

- Proof correctness of chebyshev polynomials are orthogonal:

$$\langle T_n, T_m \rangle = \int_{-1}^{1} T_n(x) \cdot T_m(x) \underbrace{\frac{1}{\sqrt{1-x^2}}}_{\to w(x)} dx$$

$$= \int_{-1}^{1} \cos(n \cdot \arccos x) \cdot \cos(m \cdot \arccos x) \frac{1}{\sqrt{1-x^2}} dx$$

$$= \int_{-1}^{1} \cos(n \cdot \arccos x) \cdot \cos(m \cdot \arccos x) \, d\arccos x$$

$$= \int_{\pi}^{0} \cos(ny)\cos(my) \, dy$$

$$= \int_{-1}^{1} \left(\cos((m-n)y) + \cos((m+n)y) \right) dy$$

$$= 0 \qquad (m \neq n).$$

∴ orthogonal!

普通多项式，x值可能特别大/小，造成数值问题

- ChebNet: (stable under perturbation of coefficients)

  → $x \in [-1, 1]$, $y \in [-1, 1]$

  - Chebyshev polynomial: $T_0 = 1,\ T_1 = x,\ T_{n+1}(x) = 2T_n(x) - T_{n-1}(x)$

    $g(\Lambda) = \theta_0 T_0(x) + \theta_1 \cdot T_1(x)$ + ...

    → Initialization for Cheb poly, $x \in [-1, 1]$.

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

    → recursive def

  ∴ $U \hat{g}(\Lambda) U^T f = \sum_{k=0}^{K} \theta_k T_k(\hat{L}) f$, where $\hat{L} = \frac{2L}{\lambda_{max}} - I$.

- GCN: set $k=1$, $\lambda_{max} = 2$ for ChebNet.

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

    Generally we want GCN be a low-pass filter for $\hat{L}$

    let $-\theta_1 = \theta_0 = \theta$ (further constrain)

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

    ∴ $U \hat{g}(\Lambda) U^T f = \theta(2I - L) f = \theta(2I - D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}) f$

    $= \theta(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) f$

    renormalization: $U \hat{g}(\Lambda) U^T f = \theta(D^{-\frac{1}{2}} \hat{A} D^{-\frac{1}{2}}) f$, where $\hat{A} = A + I$

- multi-channel GCN:

In general: $F_{out}[:, i] = \sum_{j=1}^{d_1} g_{i,j}(L) \cdot F_{in}[:, j],\ i = 1 \dots d_2$.

GCN: $F_{out}[:, i] = \sum_{j=1}^{d_1} \theta_{i,j} (D^{-\frac{1}{2}} \hat{A} D^{-\frac{1}{2}}) F_{in}[:, j]$ $i = 1 \dots d_2$

Matrix form: $F_{out} = (D^{-\frac{1}{2}} \hat{A} D^{-\frac{1}{2}}) F_{in} \cdot \theta$ $\theta \in R^{d_1 \times d_2}$.

- Spatial view for GCN:

let $C = D^{-\frac{1}{2}} \hat{A} D^{-\frac{1}{2}}$

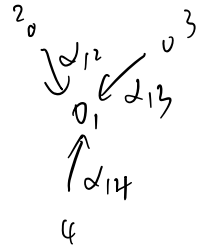
$F_{out} = C F_{in} \theta$

we know that: $C[i, j]$ if $v_j \notin N(v_i) \cup \{v_i\}$.

$\Rightarrow F_{out}[i, :] = \sum_{v_j \in N(v_i) \cup \{v_i\}} C[i, j] \cdot F_{in}[j, :] \theta$.

Aggregation. Feature transformation.

- GAT:



$$\alpha_{ij} = \frac{\exp(\sigma(a^T \cdot [Wh_i, Wh_j]))}{\sum_{k \in N_i} \exp(\sigma(a^T [Wh_i, Wh_k]))}$$

σ is LeakyRelu.

· MPNN：(考虑edge embedding)

$$m_i^{(k+1)} = \sum_{v_j \in N(v_i)} M_k(h_i^k, h_j^k, e_{ij})$$

$$h_i^{k+1} = U_k(h_i^k, m_i^{k+1}) \quad \text{, where } M_k, U_k \text{ need to be designed.}$$

· Graph Pooling：
- gPool：考虑 node importance
- DiffPool：考虑 clustering
- EigenPooling：利用滤波器 GFT，得到更好的 Hidden H_f

1. gPool：$v_i \to y_i$，$y_i = \frac{h_i^T \cdot p}{\|p\|}$ ⟶ Importance Ranking

$idx \leftarrow rank(y, n_p)$

$A_p \leftarrow A[idx, idx]$ ⟶ new Adj Matrix

$H_p \leftarrow H[idx, :] \odot \sigma(y[idx])$ ⟶ new hidden

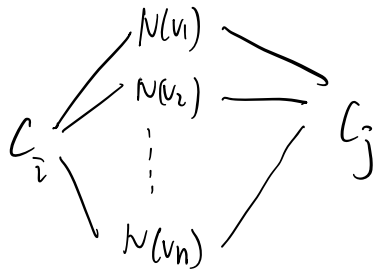
Cons：没考虑聚合性，使得 sample 的 node 连接性不强

2、DiffPool：

$H \in R^{n \times d}$ → W_1 → $H_a \in R^{n \times n_p}$ → $A_p = H_a^T A H_a$

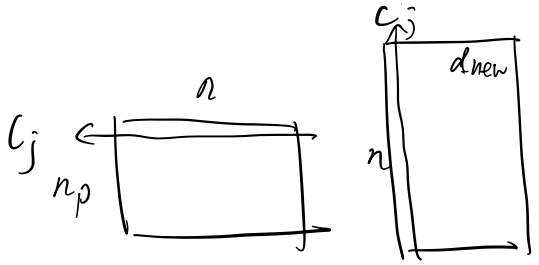
$H \in R^{n \times d}$ → W_2 → $H_f \in R^{n \times d_{new}}$ → $H_p = H_a^T H_f$

原理：1. $A_p = H_a^T A H_a$



cluster ⟷ nodes & N(v) ⟷ cluster.

2. $H_p = H_a^T H_f$



Hidden feature of C_j is from weighted sum of H_p

↓

"weight" is prob of node i & N(i) in C_j

3. Eigen pooling: Step 1: learn Ap like DiffPool

Step 2: use GFT to learn a better H_p.

subgraph G_i $\xrightarrow{GFT}$ $\hat{f}$ ⟶ Truncate some $\hat{f}_i$

*

- Eigen-decomposition is $O(N^3)$!!!

- If we want to develop a low-pass filter simply:

$$\hat{g}(\Lambda) = \begin{bmatrix} \lambda_{max} - \lambda_0 = \lambda_{max} & & & \\ & \lambda_{max} - \lambda_1 & & \\ & & \ddots & \\ & & & \lambda_{max} - \lambda_N \\ & & & \| \\ & & & 0 \end{bmatrix} \quad \begin{matrix} \leftarrow \text{low } f \\ \\ U^T f \\ \\ \leftarrow \text{high } f \end{matrix}$$

*

- Empirically LeakyRelu > Relu

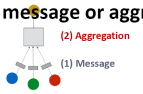
· 18 GNN 3:

- **GNN Layer:**
  - Transformation + Aggregation
  - Classic GNN layers: GCN, GraphSAGE, GAT
- **Layer connectivity:**
  - Deciding number of layers
  - Skip connections
- **Graph Manipulation:**
  - Feature augmentation
  - Structure manipulation

1. GNN Layer:

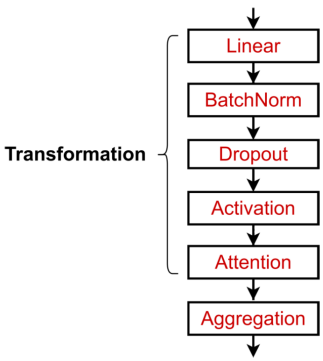
A Single GNN Layer

- **Putting things together:**
  - **(1) Message**: each node computes a mes: $\mathbf{m}_u^{(l)} = \text{MSG}^{(l)}\left(\mathbf{h}_u^{(l-1)}\right), u \in \{N(v) \cup v\}$
  - **(2) Aggregation**: aggregate messages from neighb $\mathbf{h}_v^{(l)} = \text{AGG}^{(l)}\left(\left\{\mathbf{m}_u^{(l)}, u \in N(v)\right\}, \mathbf{m}_v^{(l)}\right)$
  - **Nonlinearity (activation):** Adds expressiveness
    - Often written as $\sigma(\cdot)$: $\text{ReLU}(\cdot)$, $\text{Sigmoid}(\cdot)$, ...
    - Can be added to **message or aggregation**



DNN Tricks: (in a linear layer)

A suggested GNN Layer



2. More Layers

- Add More layers ⇒ Over Smooth Problem. (Embedding are the same for different nodes)

原因

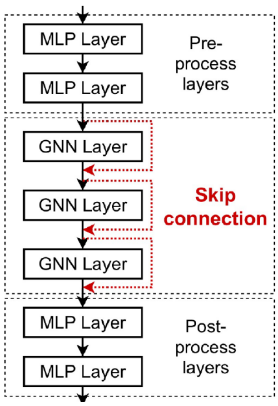
- **Receptive field:** the set of nodes that determine the embedding of a node of interest

- We knew **the embedding of a node is determined by its** receptive field
  - If two nodes **have highly-overlapped receptive fields, then their embeddings are highly similar**

- Stack many GNN layers → **nodes will have highly- overlapped receptive fields** → **node embeddings will be highly similar** → **suffer from the over-smoothing problem**

1. 假设前提1：决定合理的层数.

- **Step 1: Analyze the necessary receptive field** to solve your problem. E.g., by computing the diameter of the graph
- **Step 2:** Set number of GNN layers L to be a bit more than the receptive field we like. **Do not set L to be unnecessarily large**!



2. 改变model结构:

① Skip-Connection Formula

- **A standard GCN layer**

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

This is our $F(\mathbf{x})$

- **A GCN layer with skip connection**

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

$F(\mathbf{x})$ + $\mathbf{x}$

Other commonly used augmented features:

- **PageRank**
- **Centrality**

② 加强GNN之外的表达

Pre-processing layers: Important when encoding node features is necessary.
E.g., when nodes represent images/text

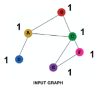
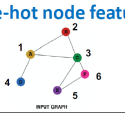
Post-processing layers: Important when reasoning / transformation over node embeddings are needed
E.g., graph classification, knowledge graphs

③ GNN layer 减少. 单个layer 加深

3. Augmentation. { Feature / Structure

- **Graph Feature manipulation**
  - The input graph **lacks features** → **feature augmentation**
- **Graph Structure manipulation**
  - The graph is **too sparse** → **Add virtual nodes / edges**
  - The graph is **too dense** → **Sample neighbors when doing message passing**
  - The graph is **too large** → **Sample subgraphs to compute embeddings**

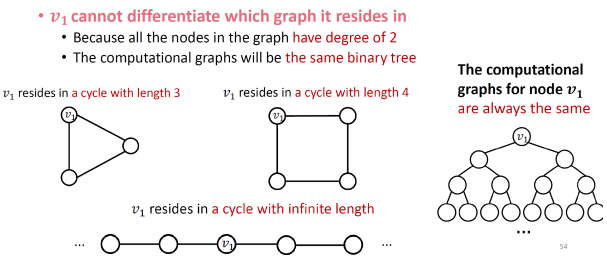
1. Feature Augment

	Constant node feature	One-hot node feature
		
Expressive power	**Medium**. All the nodes are identical, but GNN can still learn from the graph structure	**High**. Each node has a unique ID, so node-specific information can be stored
Inductive learning (Generalize to unseen nodes)	**High**. Simple to generalize to new nodes: we assign constant feature to them, then apply our GNN	**Low**. Cannot generalize to new nodes: new nodes introduce new IDs, GNN doesn't know how to embed unseen IDs
Computational cost	**Low**. Only 1 dimensional feature	**High**. O V dimensional feature, cannot apply to large graphs
Use cases	Any graph, inductive settings (generalize to new nodes)	Small graph, transductive settings (no new nodes)

Why Feature Aug?

① Input hasn't any features. more expressive (表达)

② Some structures are hard to learn



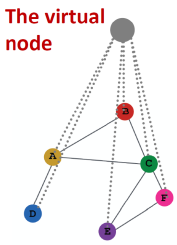
other Feature Aug Tricks: PageRank, Centrality ...

2. Structure augmentation

I. Add virtual edges: K-hops: $\tilde{A} = \sum_{k} A^k$

II. Add virtual nodes:

improve message passing in sparse graph.



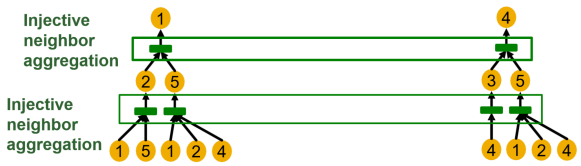
Why we need these augmentation?

- Our assumption so far has been **Raw input graph = computational graph**
- **Reasons for breaking this assumption**
  - **Feature level:**
    - The input graph **lacks features** → feature augmentation
  - **Structure level:**
    - The graph is **too sparse** → inefficient message passing
    - The graph is **too dense** → message passing is too costly
    - The graph is **too large** → cannot fit the computational graph into a GPU

29 Graph Isomorphism Network.

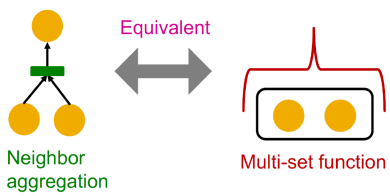
(研究的是：GNN对于structure的表达能力)

- More expressive: Injective neighbor aggregation:



Every step of aggregation is injective ⇒ GNN is injective ⇒ most expressive

- Multi-set-function:



Injective function can distinguish multi-sets!

- Fail Examples:

- **GCN** (mean-pool) [Kipf & Welling, ICLR 2017]
  - Uses **element-wise** mean pooling over neighboring node features

$$\text{Mean}(\{x_u\}_{u \in N(v)})$$

- **GraphSAGE** (max-pool) [Hamilton et al. NeurIPS 2017]
  - Uses **element-wise** max pooling over neighboring node features

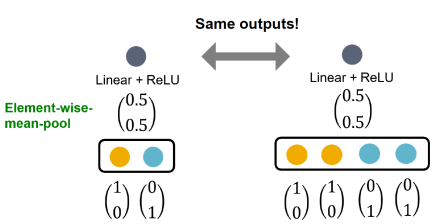
$$\text{Max}(\{x_u\}_{u \in N(v)})$$

① GCN:

Theorem [Xu et al. ICLR 2019]

- GCN's aggregation **function cannot distinguish different multi-sets with the same color proportion.**

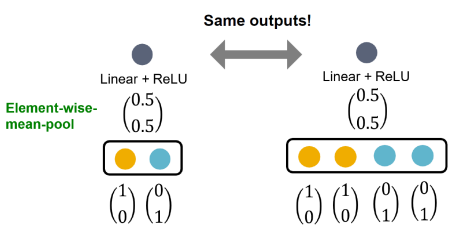
Failure case illustration



② GraphSAGE (max pool)

- GraphSAGE's aggregation **function cannot distinguish different multi-sets with the same set of distinct colors.**

Failure case illustration



· Injective Multi-set function:

· Thm: Injective function can be expressed as: $\Phi\left(\sum_{x \in S} f(x)\right)$

non-linear; linear; multi-set.

· How to model Φ, f? MLP.

Thm (Universal Approximation Thm):

MLP(hidden = 1) with $\sigma(\cdot)$ can approximate any continuous function.

$\Rightarrow$ Injective function $\Leftrightarrow$ $MLP_\Phi\left(\sum_{x \in S} MLP_f(x)\right)$

· GIN: $MLP_\Phi\left(\sum_{x \in S} MLP_f(x)\right)$

· Thm: GIN's aggregation is injective.

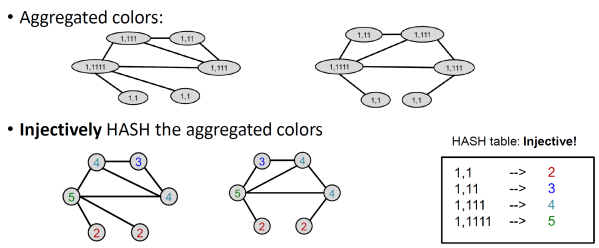
$\Rightarrow$ GIN is most expressive!

· WL graph kernal:

· Checking isomorphism is NP-hard!

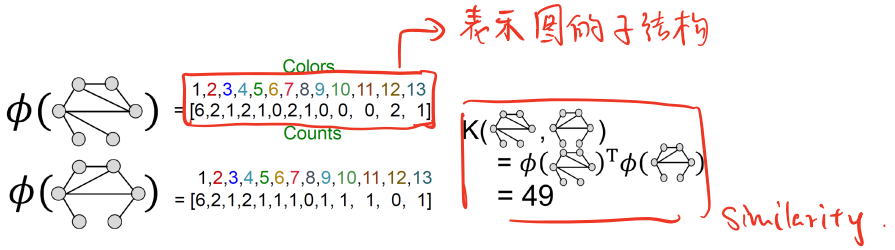
· WL kernal can distinguish most of the real-world graphs.

· Color refinement: $c^{(k+1)}(v) = HASH\left(c^{(k)}(v), \{c^{(k)}(u)\}_{u \in N(v)}\right)$



HASH is injective!

· Iterate till: no difference between nodes in a Graph in iteration $i+1$.



- KL kernel can somehow show the similarity.
  However $\phi(G_1) = \phi(G_2)$ doesn't imply isomorphism!

- $O(|E|)$ algorithm.
  (since there are $2|E|$ multiplication for each iteration)

- GIN model: use NN to model the HASH function
  - $MLP_\phi\left((1+\varepsilon)\cdot MLP_f(c^{k}(v)) + \sum_{u\in N(v)} MLP_f(c^{(k)}(u))\right)$
  - Since MLP can provide "one-hot" input for next layer:

$$GINConv\left(c^{k}(v), \{c^{k}(u)\}_{u\in N(v)}\right) = MLP_\phi\left((1+\varepsilon)\cdot c^{k}(v) + \sum_{u\in N(v)} c^{(k)}(u)\right)$$

- Iteratively update node vectors by (Differentiable color HASH function)

$$c^{(k+1)}(v) = \text{GINConv}\left(\{c^{(k)}(v), \{c^{(k)}(u)\}_{u\in N(v)}\}\right),$$

- **GIN can be understood as differentiable neural version of the WL graph Kernel:**

	Update target	Update function
WL Graph Kernel	Node colors (one-hot)	HASH
GIN	Node embeddings (low-dim vectors)	GINConv

↳ Capture fine-grained similarity of different nodes.

- Rank of expression: Sum > mean > max

L10 Heterogeneous Graphs, Knowledge Graphs.

1. Heterogeneous Graphs

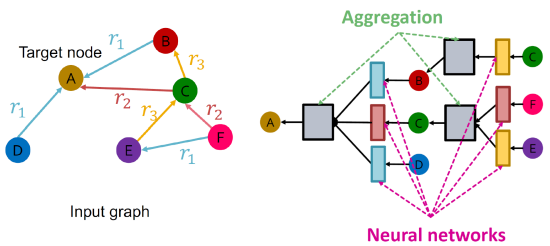
A heterogeneous graph is defined as

- $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{R}, \boldsymbol{T})$

- Nodes with node types $v_i \in V$
- Edges with relation types $(v_i, r, v_j) \in E$
- Node type $T(v_i)$
- Relation type $r \in R$

· Method: Relational GCN.

- Use different neural network weights for different relation types.



$$h_v^{(l+1)} = \sigma\left(\sum_{r \in R} \sum_{u \in N_v^r} \frac{1}{c_{v,r}} W_r^{(l)} h_u^{(l)} + W_0^{(l)} h_v^{(l)}\right)$$

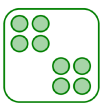
· params: $\{W_r^1, \cdots W_r^L\}_{r \in R}$.

⇒ too many params, overfitting

· Solution:

① block diagonal Matrix:

- **Key insight**: make the weights **sparse**!
- Use **block diagonal matrices** for $\mathbf{W}_r^{(l)}$

$\mathbf{W}_r =$ 

Limitation: only nearby neurons/dimensions can interact through W

- If use B low-dimensional matrices, then # param reduces from $d^{(l+1)} \times d^{(l)}$ to $B \times \frac{d^{(l+1)}}{B} \times \frac{d^{(l)}}{B}$

② Basis learning:

$W_r = \sum_{b=1}^{B} a_{rb} \cdot V_b$ → basis matrix; a_{rb} → importance.

2. knowledge Graph

- KG completion : given (h, r) , predict tails.

  that is $\arg\max_{\theta} \prod_{i,j,k \in N} P(X_{ijk}=1 \mid \theta, G)$, $X_{ijk}=1$ means $i \Rightarrow k$ is plausible.

- Methods:

  - Path Rank Alg: (Random Walk)

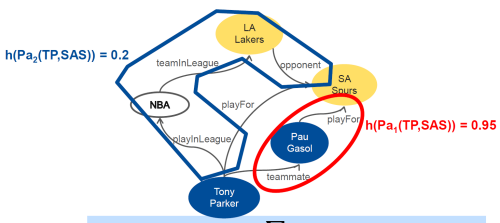
    $$S(A,r,B) = \sum_{p \in paths(A \to B)} \theta_r^p \cdot Pr(p)$$

    $$rel = \arg\max_{r \in rels} S(A,r,B)$$

    $$obj = \arg\max_{e \in ents} S(A,R,e)$$

Path Ranking Algorithm [Lao et al., 11]

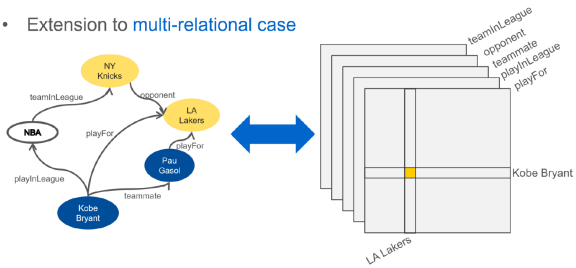
- Random walks on the graph are used to sample paths
- Paths are weighted with probability of reaching target from source
- Paths are used as ranking experts in a scoring function



$$S(TP, playFor, SAS) = \sum_{i \in paths} \theta^i_{playFor} h(pa_i(TP, SAS))$$

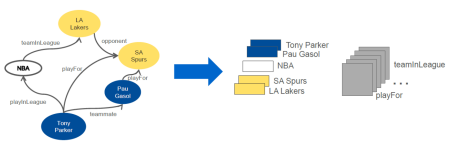
  - Factorization:

    - Extension to multi-relational case



  - Embedding:

    - Related to Deep Learning methods
    - Entities are vectors (low-dimensional sparse)
    - Relation types are operators on these vectors



Embeddings trained to define a **similarity score** on triples such that

$$S(KB, playFor, LAL) > S(KB, playFor, NYK)$$

- KG Embeding : (h, r, t).

  Goal : (h, r) ≈ t

Relation Patterns

- **Symmetric (Antisymmetric) Relations:** $r(h,t) \Rightarrow r(t,h)$ $(r(h,t) \Rightarrow \neg r(t,h))$ $\forall h,t$
  - **Example**:
    - Symmetric: Family, Roommate
    - Antisymmetric: Hypernym
- **Inverse Relations:** $r_2(h,t) \Rightarrow r_1(t,h)$
  - **Example** : (Advisor, Advisee)
- **Composition (Transitive) Relations:** $r_1(x,y) \wedge r_2(y,z) \Rightarrow r_3(x,z)$ $\forall x,y,z$
  - **Example**: My mother's husband is my father.
- **1-to-N relations:** $r(h,t_1), r(h,t_2), \ldots, r(h,t_n)$ are all True.
  - **Example**: r is "StudentsOf"

1. TransE: $h + r \approx t$

Score: $f_r(h,t) = -d_r(h,t) = -\|h+r-t\|$

Loss function: $L(h,r,t) = \max(0, \gamma + d_{r,pos} - d_{r,neg})$

$= \max(0, \gamma + f_{r,neg} - f_{r,pos})$

Algorithm 1 Learning TransE

input Training set $S = \{(h,\ell,t)\}$, entities and rel. sets E and L, margin γ, embeddings dim. k.

1: **initialize** $\ell \leftarrow$ uniform$(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each $\ell \in L$
2: $\ell \leftarrow \ell / \|\ell\|$ for each $\ell \in L$
3: $\mathbf{e} \leftarrow$ uniform$(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each entity $e \in E$

Entities and relations are initialized uniformly, and normalized

4: **loop**
5: $\mathbf{e} \leftarrow \mathbf{e} / \|\mathbf{e}\|$ for each entity $e \in E$
6: $S_{batch} \leftarrow$ sample(S, b) // sample a minibatch of size b
7: $T_{batch} \leftarrow \emptyset$ // initialize the set of pairs of triplets
8: **for** $(h,\ell,t) \in S_{batch}$ **do**
9: $(h',\ell,t') \leftarrow$ sample$(S'_{(h,\ell,t)})$ // sample a corrupted triplet

Negative sampling with triplet that does not appear in the KG

10: $T_{batch} \leftarrow T_{batch} \cup \{((h,\ell,t),(h',\ell,t'))\}$
11: **end for**
12: Update embeddings w.r.t. $\sum_{((h,\ell,t),(h',\ell,t')) \in T_{batch}} \nabla[\gamma + d(\boldsymbol{h}+\boldsymbol{\ell},\boldsymbol{t}) - d(\boldsymbol{h'}+\boldsymbol{\ell},\boldsymbol{t'})]_+$

d represents distance (negative of score)

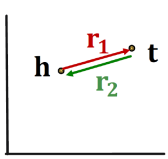
positive sample / negative sample

13: **end loop**

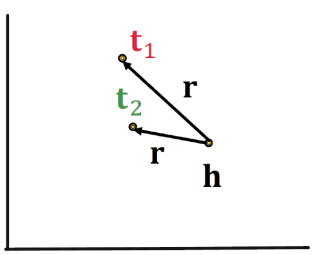
Contrastive loss: favors lower distance (or higher score) for valid triplets, high distance (or lower score) for corrupted ones

Inverse (✓)

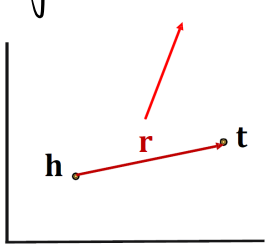
$\mathbf{h} + \mathbf{r}_2 = \mathbf{t}$, we can set $\mathbf{r}_1 = -\mathbf{r}_2$



1-to-N (✗)

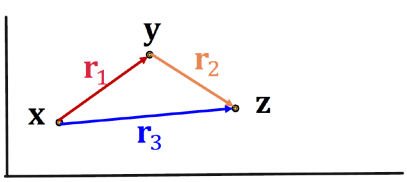


Symmetric (✗)

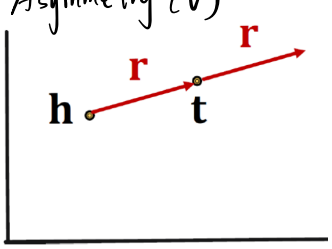


Composition (✓)

- $\mathbf{r}_1 = \mathbf{r}_2 + \mathbf{r}_3$

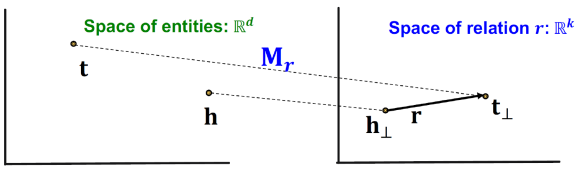


Asymmetry (✓)



2. TransR: entity space $\mathbb{R}^d$ $\xrightarrow{M_r \in \mathbb{R}^{k \times d}}$ relation space $\mathbb{R}^k$

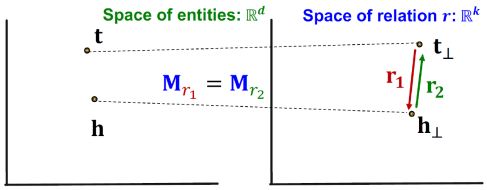
- $h_\perp = M_r h$, $t_\perp = M_r t$
- Score: $-\|h_\perp + r - t_\perp\|$



Inverse: ✓

$$\mathbf{r}_2 = -\mathbf{r}_1, \mathbf{M}_{r_1} = \mathbf{M}_{r_2}$$

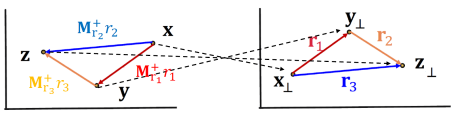
$$\mathbf{M}_{r_1}\mathbf{t} + \mathbf{r}_1 = \mathbf{M}_{r_1}\mathbf{h} \quad \mathbf{M}_{r_2}\mathbf{h} + \mathbf{r}_2 = \mathbf{M}_{r_2}\mathbf{t}$$



Composition ✓

- TransR can model composition relations ✓

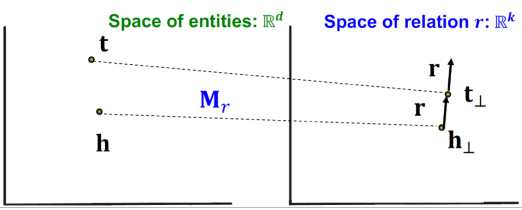
$\mathbf{M}_{r_i}^+ = \mathbf{M}_{r_i}^T(\mathbf{M}_{r_i}\mathbf{M}_{r_i}^T)^{-1}$ is the Moore–Penrose inverse (pseudoinverse) of $\mathbf{M}_{r_i}$ where $\mathbf{M}_{r_i}\mathbf{M}_{r_i}^T$ is invertible (full rank); so that $\mathbf{M}_{r_i}\mathbf{M}_{r_i}^+ = I$



Asymmetry: ✓

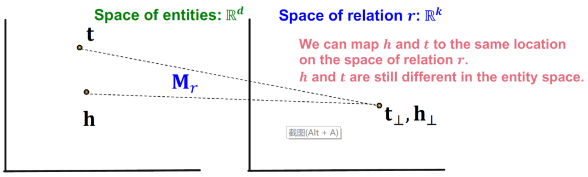
$$\mathbf{r} \neq 0, \mathbf{M}_r\mathbf{h} + \mathbf{r} = \mathbf{M}_r\mathbf{t},$$

Then $\mathbf{M}_r\mathbf{t} + \mathbf{r} \neq \mathbf{M}_r\mathbf{h}$ ✓



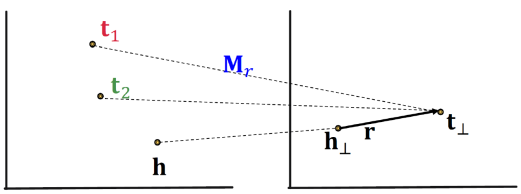
Symmetric: ✓

$$\mathbf{r} = 0,\ \mathbf{h}_\perp = \mathbf{M}_r\mathbf{h} = \mathbf{M}_r\mathbf{t} = \mathbf{t}_\perp$$ ✓



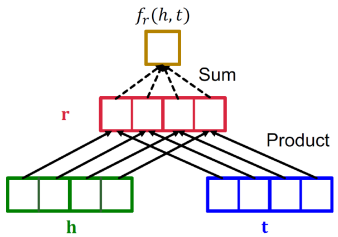
1-to-N: ✓

- We can learn $\mathbf{M}_r$ so that $\mathbf{t}_\perp = \mathbf{M}_r\mathbf{t}_1 = \mathbf{M}_r\mathbf{t}_2$
- Note that $\mathbf{t}_1$ does not need to be equal to $\mathbf{t}_2$!



3. DistMult: (bilinear Modeling)

- $h, r, t \in \mathbb{R}^k$
- score: $f_r(h,t) = \langle h, r, t \rangle = \sum_i h_i \cdot r_i \cdot t_i$
- Intuition: dot product of $h \cdot r$ and t.



Symmetric: (✓)

DistMult can naturally model symmetric relations

$$f_r(h,t) = \langle \mathbf{h}, \mathbf{r}, \mathbf{t} \rangle = \sum_i \mathbf{h}_i \cdot \mathbf{r}_i \cdot \mathbf{t}_i = \langle \mathbf{t}, \mathbf{r}, \mathbf{h} \rangle = f_r(t,h)$$

Inverse (X)

DistMult cannot model inverse relations

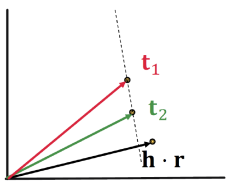
- If it does model inverse relations:

$$f_{r_2}(h,t) = \langle \mathbf{h}, \mathbf{r}_2, \mathbf{t} \rangle = \langle \mathbf{t}, \mathbf{r}_1, \mathbf{h} \rangle = f_{r_1}(t,h)$$

- This means $r_1 = r_2$

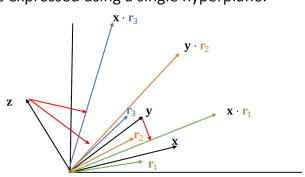
1-to-N (✓)

$$\langle \mathbf{h}, \mathbf{r}, \mathbf{t}_1 \rangle = \langle \mathbf{h}, \mathbf{r}, \mathbf{t}_2 \rangle$$



Composition (X)

- **Intuition:** **DistMult** defines a hyperplane for each (head, relation), the union of the hyperplanes induced by multihops of relations, e.g., (r_1, r_2), cannot be expressed using a single hyperplane.



5 Embedding Alg:

Model	Sym.	Antisym.	Inv.	Compos.	1-to-N
TransE	✗	✓	✓	✓	✗
TransR	✓	✓	✓	✓	✓
DistMult	✓	✗	✗	✗	✓
ComplEx	✓	✓	✓	✗	✓
RotatE	✓	✓	✓	✓	✓

L10.11 Recommendation System.

1. Collebrative Filetering (CF)

- input : user-item matrix.
- output : recommended items.

缺点：

- A popular similarity measure in user-based CF: **Pearson correlation**
  - a, b : users
  - $r_{a,p}$: rating of user a for item p
  - P : set of items, rated both by a and b
  - $sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$
- Possible similarity values between −1 and 1

	Item1	Item2	Item3	Item4	Item5	
Alice	5	3	4	4	?	
User1	3	1	2	3	3	sim = 0.85
User2	4	3	4	3	5	sim = 0.70
User3	3	3	1	5	4	sim = 0.00
User4	1	5	5	2	1	sim = -0.79

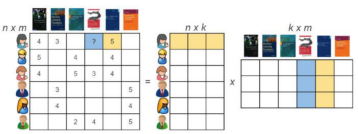
17

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} |sim(a,b)|}$$

优点：

- Matrix Factorization is one of the most popular methods for collaborative filtering
  - Given rating matrix Y
  - Each row represents an user u
  - While each column an item i

Regularization of p and q should be applied



$$\hat{y}_{ui} = f(u,i|\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^T \mathbf{q}_i = \sum_{k=1}^{K} p_{uk} q_{ik}$$

$$L_{sqr} = \sum_{(u,i) \in \mathcal{Y} \cup \mathcal{Y}^-} w_{ui}(y_{ui} - \hat{y}_{ui})^2$$

25

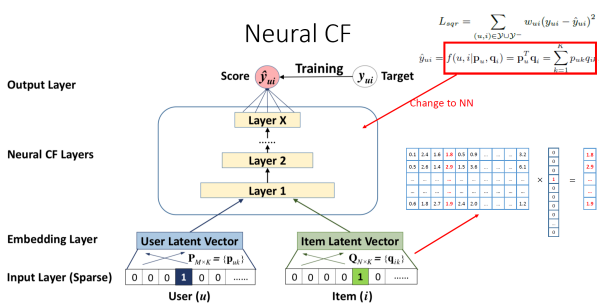
Metric:

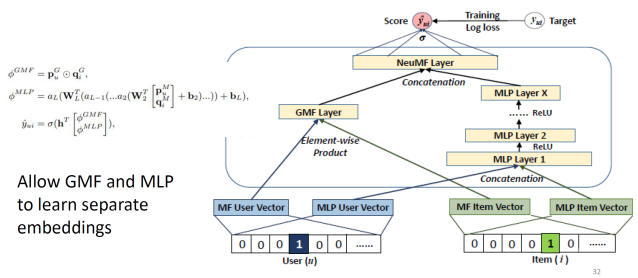
$$MAE = \frac{1}{n}\sum_{i=1}^{n}|p_i - r_i|$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - r_i)^2}$$

- Pros:
  - well-understood, works well in some domains, no knowledge engineering required
- Cons:
  - requires user community, sparsity problems, no integration of other knowledge sources, no explanation of results

2. Neural Collaborative Filtering (NLF)

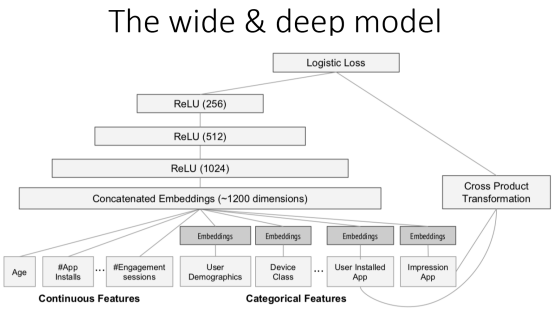




Allow GMF and MLP to learn separate embeddings

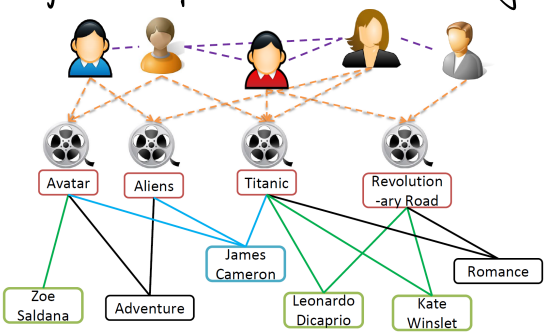
32

3. Wide. Deep learning



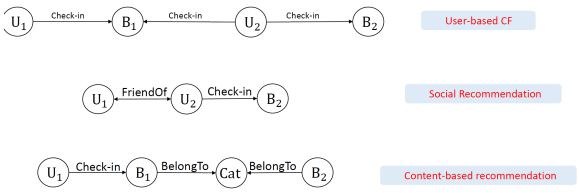
$$P(Y=1|\mathbf{x}) = \sigma(\mathbf{w}_{wide}^T[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{deep}^T a^{(l_f)} + b)$$

4. Heterogeneous Info for Recommendation System.

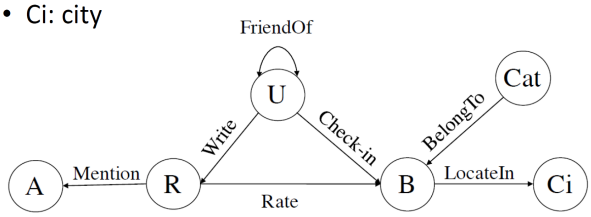


Metapath based RS

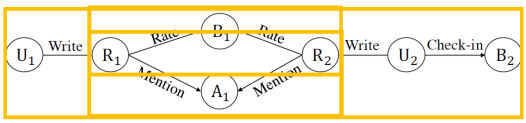
- Metapath → Recommending Strategy.



- R: reviews;
- U: users;
- B: business;
- Cat: category of item;
- Ci: city



Compute Similarity based on a Meta-graph



Compute $\mathbf{C}_{P_1}$: $\mathbf{C}_{P_1} = \mathbf{W}_{RB} \cdot \mathbf{W}_{RB}^\top$;
Compute $\mathbf{C}_{P_2}$: $\mathbf{C}_{P_2} = \mathbf{W}_{RA} \cdot \mathbf{W}_{RA}^\top$;
Compute $\mathbf{C}_{S_r}$: $\mathbf{C}_{S_r} = \mathbf{C}_{P_1} \odot \mathbf{C}_{P_2}$;
Compute $\mathbf{C}_{M_9}$ $\mathbf{C}_{M_9} = \mathbf{W}_{UR} \cdot \mathbf{C}_{S_r} \cdot \mathbf{W}_{UR}^\top \cdot \mathbf{W}_{UB}$;

L12 Recommendation Sys 2.

· Problem: MF doesn't explicitly use graph structure.

⇒ Only First-order graph structure capture in training objective

· Solution : GNN (explicitly, high order)

1. NGCF : NGCF jointly learn :

1. GNN 2. node shallow embedding

- Iteratively update node embeddings using neighboring embeddings.

$$\boldsymbol{h}_v^{(k+1)} = \text{COMBINE}\left(\boldsymbol{h}_v^{(k)}, \text{AGGR}\left(\left\{\boldsymbol{h}_u^{(k)}\right\}_{u \in N(v)}\right)\right)$$

$$\boldsymbol{h}_u^{(k+1)} = \text{COMBINE}\left(\boldsymbol{h}_u^{(k)}, \text{AGGR}\left(\left\{\boldsymbol{h}_v^{(k)}\right\}_{v \in N(u)}\right)\right)$$

$\sigma(W \cdot [x, y])$ Mean

· Finally : At k^{th} iteration :

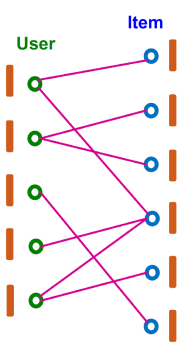
- For all $u \in \boldsymbol{U}$, $v \in \boldsymbol{V}$, we set

$$\boldsymbol{u} \leftarrow \boldsymbol{h}_v^{(K)}, \ \boldsymbol{v} \leftarrow \boldsymbol{h}_v^{(K)}.$$

- Score function is the inner product:

$$\text{Score}(u, v) = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{v}$$

· User - item - User : very informative



2. LightGCN.

- Shallow embedding : $O(ND)$
- GNN : $O(D^2)$ $\quad f: R^D \to R^D$

- Simplifying GCN

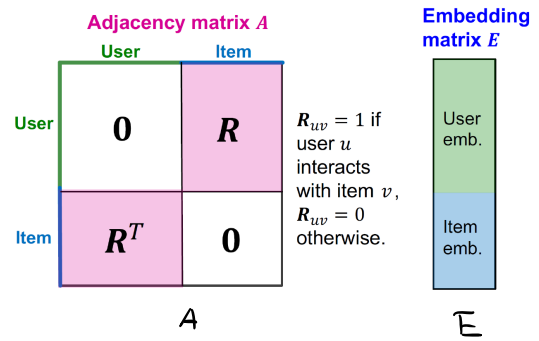
- GCN: $E^{(k+1)} = \sigma(\tilde{A} E^{(k)} W^{(k)})$, $\tilde{A} = D^{-1/2} A D^{-1/2}$

- Simplify GCN by removing ReLU:

$$E^k = \tilde{A} E^{k-1} W^{k-1}$$
$$= \tilde{A}(\tilde{A} E^{k-2}) W^{k-2} W^{k-1}$$
$$\cdots$$
$$= \tilde{A}^k E^0 (W^0 \cdots W^{k-1})$$

($W^0 \cdots W^{k-1} = W$)

$$\Rightarrow E^k = \hat{A}^k E W$$

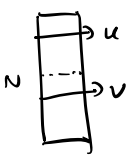


- Alg: $E \leftarrow \hat{A} E$ for every k.

- Final E: $E_{final} = \alpha_0 E^0 + \alpha_1 E^1 + \cdots + \alpha_k E^k \quad \left(\alpha = \frac{1}{k+1}\right)$

$$= \frac{1}{k+1} \sum_k \alpha_k E^k$$

- E_{final} for score: $u^T v \quad (u, v \in E)$



- Remember: Normalized Laplace: $D^{-1}(D-W)$, $D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}$

$$\sum_{k=0}^{\infty} \tilde{A}^k = (I - \tilde{A})^{-1} = \left(D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}\right)^{-1} = L^{-1}$$

LightGCN

$L = U \Lambda U^T$, $L^{-1} = U \Lambda^{-1} U^T$ ($Lu = \lambda u \Rightarrow L^{-1} L u = \lambda L u \Rightarrow L^{-1} u = \frac{1}{\lambda} u$)

Low-pass filter

∴ LightGCN can be thought as a low-pass filter. $\left(\sum_k A^k\right) \cdot E = L^{-1} E$

- Simplification leads to better performance. (than NGCF)

3. PinSAGE:

- Learn embeddings z_i: $d(z_{cake1}, z_{cake2}) < d(z_{cake}, z_{sweater})$

- Share Negative Sample: V_{neg} for a batch U_{mini} (performance stay similar!)

- Curriculum learning : make negative gradually harder
    - At n-th epoch , Add n-1 hard negative items
- Hard Negatives : close but not connected!
    - method: Run PageRank , random sample from high ranking item (but not too high)