Data Mining:

Advanced Techniques

Graph Mining: part 3

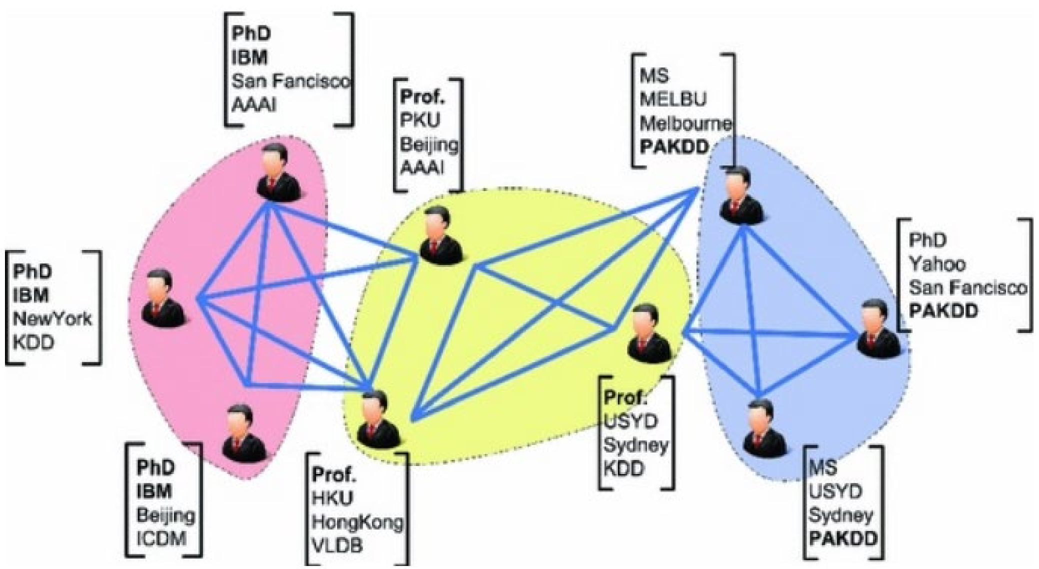
- 主讲教师: 陈佳伟, sleepyhunt@zju.edu.cn
  - https://jiawei-chen.github.io/
- TA: 陈思睿, chenthree@zju.edu.cn

Slides from Sheng Zhou

https://zhoushengisnoob.github.io/

Review --- Graph data

- A graph is a mathematical structure used to model pairwise relations between objects.
- Graph can be formally defined as: $G = \{V, E, X\}$
  - V denotes the set of nodes
  - E denotes the set of edges between nodes
  - X denotes the set of features of nodes

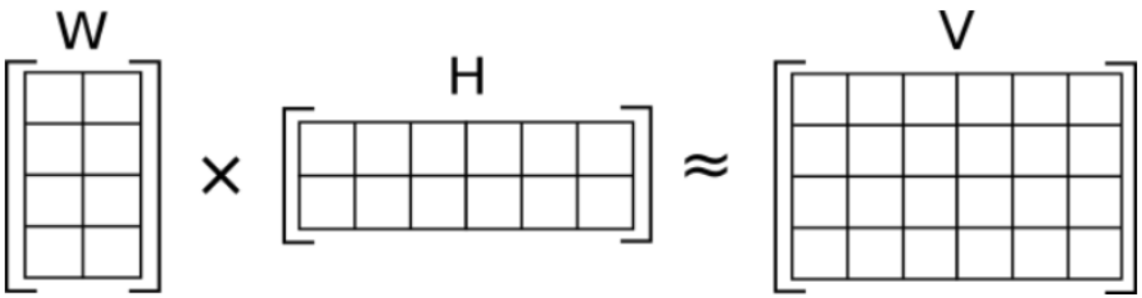


Review---Matrix factorization

- Matrix factorization
  - factorizing a matrix into a product of two lower-dimensional matrices
  - The goal is to approximate the original matrix by capturing its underlying structure and patterns

W, H: embedding of nodes

V: graph features



Slide from Sheng Zhou

Review---Random walk

- Node representation via random walk
  - Performing random walk on graph to generate multiple sequences of nodes
  - utilizing word2vec technique to get the node representation

Slide from Sheng Zhou

Review--- Graph Neural Network

- Graph data
- Classic graph representation learning
- Graph neural network
  - Three perspectives
  - Applications
  - Promising directions

Review--- GNN from Spatial Perspective

- Formal definition

$$\begin{aligned}\mathbf{h}_u^{(k+1)} &= \mathrm{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)},\ \mathsf{AGGREGATE}^{(k)}\left(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\}\right)\right) \\ &= \mathrm{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right),\end{aligned}$$

$$\mathbf{h}_u^{(0)} = \mathbf{x}_u \qquad \mathbf{z}_u = \mathbf{h}_u^{(K)}, \forall u \in \mathcal{V}$$

- Two key steps:
  - Aggregate: how to aggregate information from neighbors
  - Update: how to update the node features based on the information from the neighbors

Review--- GNN from Spectral Perspective

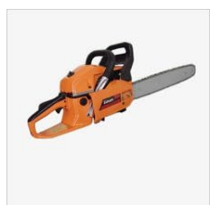
- Summary of GNN from spectral perspective
  - Convolutional network (Why, How, transfer to graph)
  - Challenge in graph (undefined)
  - Spectral --- Fourier Transform (basis is important)
  - Basis in graph (what we want? Smoothing!)
  - Define Laplace matrix in graph (estimate smoothing)
  - Basis --- eigenvector of Laplace matrix
  - Different strategies of filtering (efficient!)
  - GCN --- simplified but deep!

Review--- GNN from Spatial Perspective

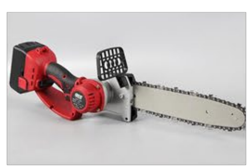
怎么找idea? --- 更好的模型(借鉴)

现有方法:

观察到:



新的idea:



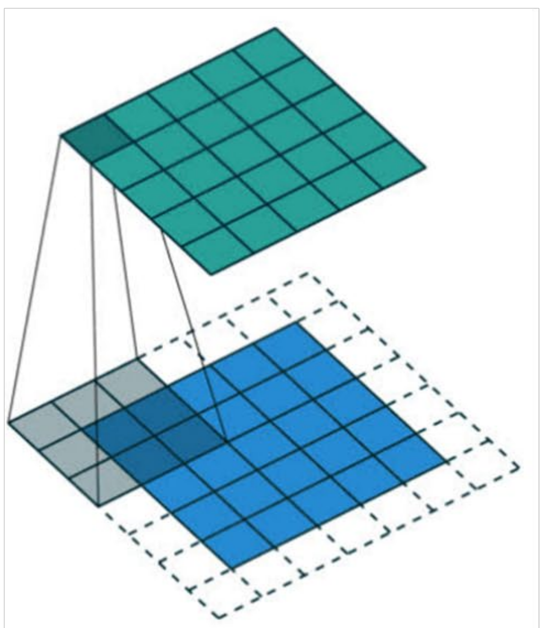
为何方法更
适合这个问题?

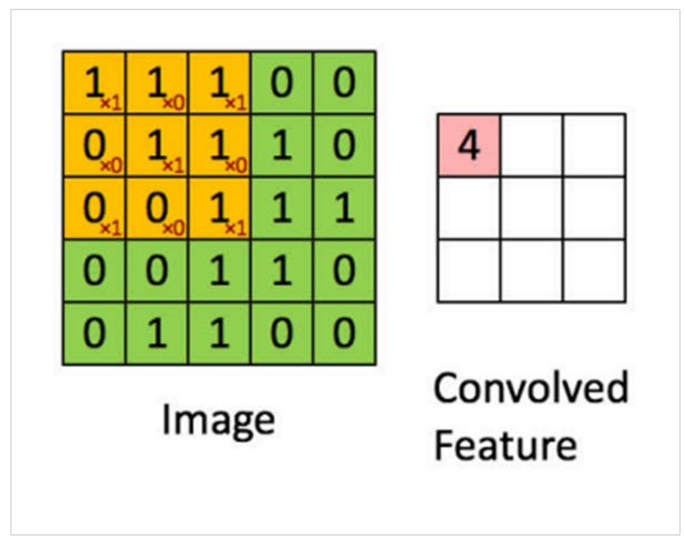
Review--- GNN from Spatial Perspective

Have seen:

- Convolutional network in computer vision

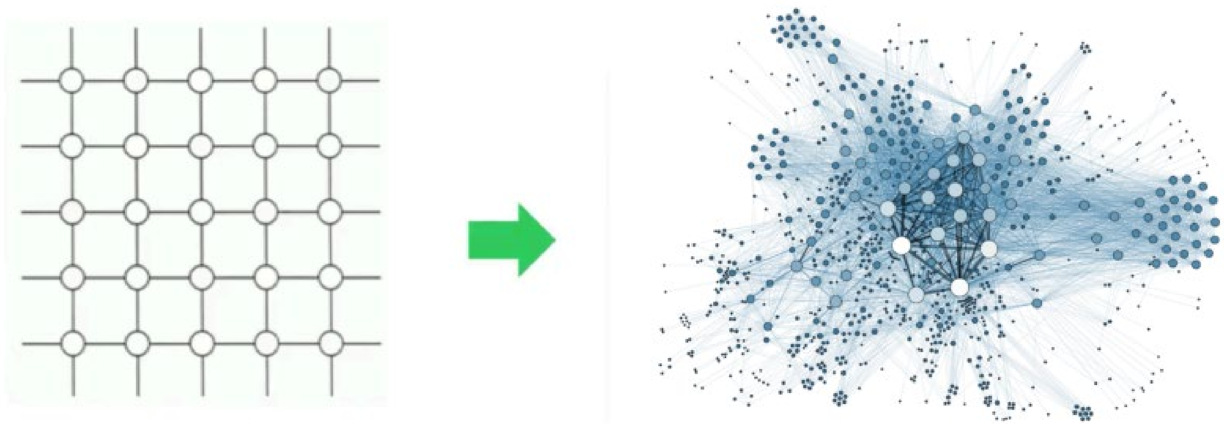
$$h(x,y) = (f * g)(x,y) = \sum_{m,n} f(x-m, y-n) g(m,n)$$





Review--- GNN from Spatial Perspective

Challenges:

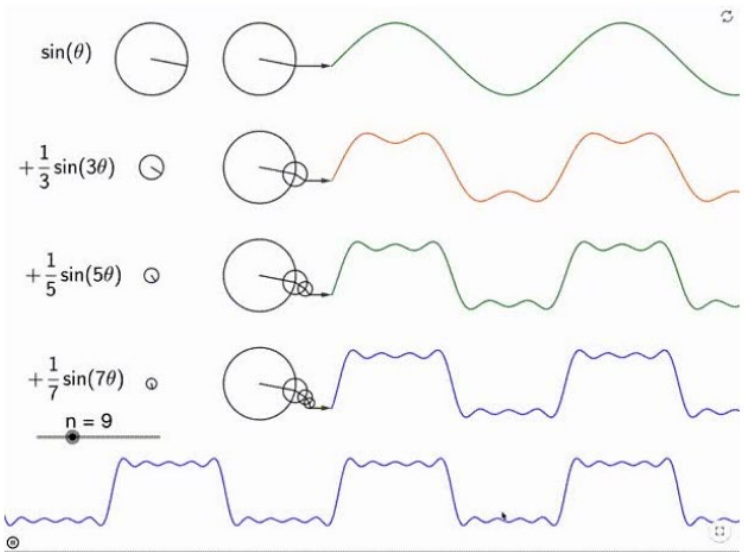


Grid-like network　　　　Irregular network

- Permutation invariance
- Undefination of f(x,**y-1**)?
- Varying number of neighbors
  - hard to define convolution kernel

Review--- GNN from Spatial Perspective

Solutions: Spectral! Define Fourier Transform on Graph



信号的变换

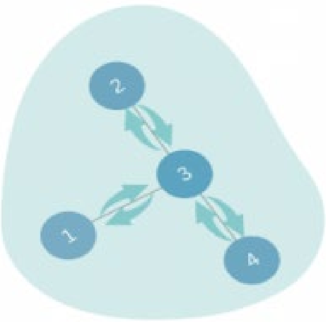
- Disentangle the original signals according to the frequencies
  Different level of smoothing

- Facilitate convolution

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}$$

Review--- GNN from Spatial Perspective

Solutions: Spectral! Define Fourier Transform on Graph



- Define basis from Laplace matrix
  - $[u_1, u_2, u_3, \dots, u_{\mathrm{n}}]$ are the eigenvectors of the matrix L
  - u_{k} indicates different level of smoothing, $\lambda_k = u_k^T L u_k$
  - Capturing graph structure: $L = \mathrm{D} - \mathrm{A}$
  - Orthogonal
  - Easy calculation
- Fourier transformation on graph

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

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

Review--- GNN from Spatial Perspective

GNN：Define various convolutional kernel

$$f * g = U\left(\left(U^T f\right) \cdot \left(U^T g\right)\right)$$
$$f * g = U g_\theta U^T f$$

According to practical needs:

$$g_\theta(\Lambda) = \mathrm{diag}(\theta)$$

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k,$$

Review--- GNN from Loss Perspective

GCN optimizes smoothing regularizer

Considering the following objective function:

$$\arg\min_{\mathbf{F}} \mathcal{L} = ||\mathbf{F} - \mathbf{S}||_F^2 + c \cdot tr(\mathbf{F}^\top \mathbf{L}\mathbf{F}).$$

$$\mathbf{F}^{(k)} \approx \theta \tilde{A} \mathbf{F}^{(k-1)}$$

GCN is equivalence to optimizing this formula via gradient descent

Complex Graph Learning

- Complex Graph
  - Directed Graph
  - Heterogeneous Graph
  - Dynamic Graph
- Challenges:
  - Complex relations, Different Structures

Complex Graph Mining

- Complex Graph
  - Directed Graph
  - Heterogeneous Graph
  - Dynamic Graph
- Challenges:
  - Complex relations, Unique Nature

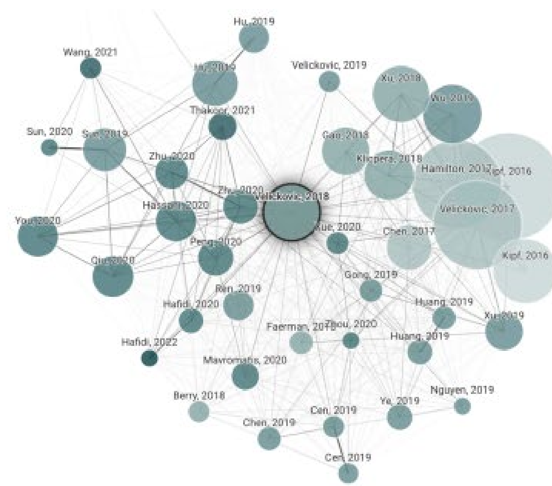
Complex Graph Mining

- Directed Graph
  - A Directed Network/Graph refers to a network where edges have **specific directions.**
  - In directed networks, edges not only contain adjacency information but often also encompass **hierarchical or semantic information**.

Complex Graph Mining

- Directed Graph
  - More common in real-world
    - I love you does not suggest you love me….

社交网络



引用网络

Complex Graph Mining

- Challenge of directed graph learning
  - Asymmetric Proximity/Relations
- Key research topics:
  - How to capture asymmetric proximity?
  - How to capture (Local) hierarchical structure?

Complex Graph Mining-Directed Graph

- Build on the foundation of vanilla graph model

1. Matrix Factorization Based
   1. HOPE[Ou et al., 2016]
2. Random Walk Based
   1. APP[Zhou et al., 2017]
   2. InfoWalk[Zhou et al., 2021]
3. Graph Neural Networks Based

Complex Graph Mining-Directed Graph

- Matrix Factorization-Based Methods:

$$\min \|\mathbf{S} - \mathbf{U}^s \cdot {\mathbf{U}^t}^\top\|_F^2$$

$$\mathbf{S} = \mathbf{M}_g^{-1} \cdot \mathbf{M}_l$$

Two sets of Embeddings

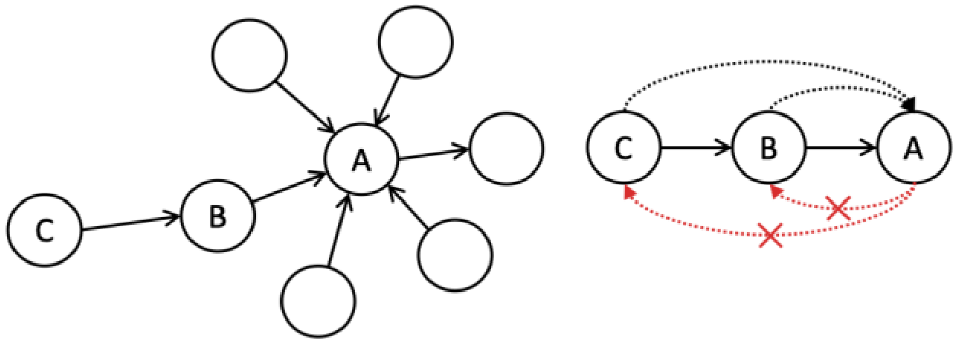
Table 1: General Formulation for High-order Proximity Measurements

Proximity Measurement	$\mathbf{M}_g$	$\mathbf{M}_l$
Katz	$\mathbf{I} - \beta \cdot \mathbf{A}$	$\beta \cdot \mathbf{A}$
Personalized Pagerank	$\mathbf{I} - \alpha\mathbf{P}$	$(1-\alpha) \cdot \mathbf{I}$
Common neighbors	$\mathbf{I}$	$\mathbf{A}^2$
Adamic-Adar	$\mathbf{I}$	$\mathbf{A} \cdot \mathbf{D} \cdot \mathbf{A}$

Asymmetric Transitivity Preserving Graph Embedding(KDD 2016)

Complex Graph Mining-Directed Graph

- Random-walk-based Methods-APP:



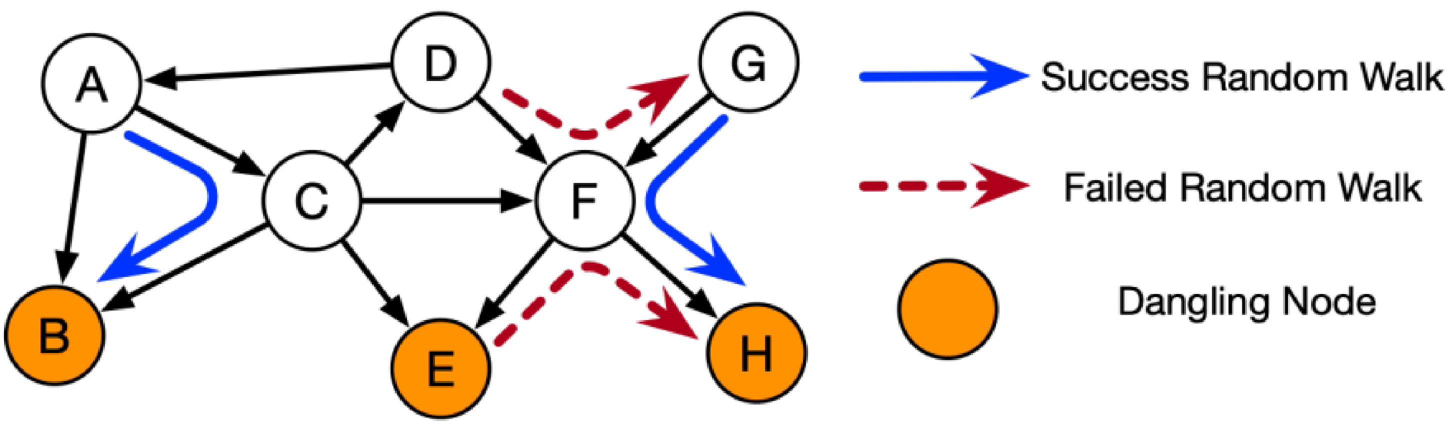
Asymmetric Proximity[2]

$$\ell = \sum_{u} \sum_{v} \#Sampled_u(v) \cdot (log\sigma(\vec{s_u} \cdot \vec{t_v})$$
$$+k \cdot E_{t_n \sim P_D}[log\sigma(-\vec{s_u} \cdot \vec{t_n})])$$

Scalable Graph Embedding for Asymmetric Proximity (AAAI'17)

Complex Graph Mining-Directed Graph

- Random-walk-based Methods-infowalk:



有向网络中的随机游走

Direction-Aware User Recommendation Based on Asymmetric Network Embedding (TOIS 2021)

Complex Graph Mining-Directed Graph

- Random-walk-based Methods-infowalk:
  - Ignoring the direction of edges during random walk.

$$P\left(\mathcal{R}_{v_i}^{k+1}=b \mid \mathcal{R}_{v_i}^{k}=a\right)=\begin{cases}\frac{1}{d_a^{\text{out}}+d_a^{\text{in}}} & \mathrm{E}_{ab}=1 \text{ or } \mathrm{E}_{ba}=1 \\ 0 & \text{otherwise}\end{cases}$$

  - The direction of edges is recorded.

$$r_{i,i+1}=\begin{cases}1 & \text{if } \mathrm{E}_{i,i+1}=1 \text{ and } \mathrm{E}_{i+1,i}=0 \\ -1 & \text{if } \mathrm{E}_{i,i+1}=0 \text{ and } \mathrm{E}_{i+1,i}=1 \\ 0 & \text{if } \mathrm{E}_{i,i+1}=1 \text{ and } \mathrm{E}_{i+1,i}=1\end{cases}$$

Direction-Aware User Recommendation Based on Asymmetric Network Embedding (TOIS 2021)

Complex Graph Mining-Directed Graph

- Random-walk-based Methods-infowalk:
  - The random walk results in a sequence with directions.

$$v_i \xrightarrow{r_{i,j}} v_j \xrightarrow{r_{j,j+1}} \dots \xrightarrow{r_{k-1,k}} v_k$$

  - Direction information is encoded into learning:

$$s_{i,i+k} = \frac{1}{k} \sum_{j=i}^{i+k-1} r_{j,j+1}$$

$$P\left(v \mid u, s_{u,v} > 0\right) = \frac{\exp\left(h_u^s \cdot h_v^t\right)}{\sum_{k \in V} \exp\left(h_u^s \cdot h_v^t\right)}$$

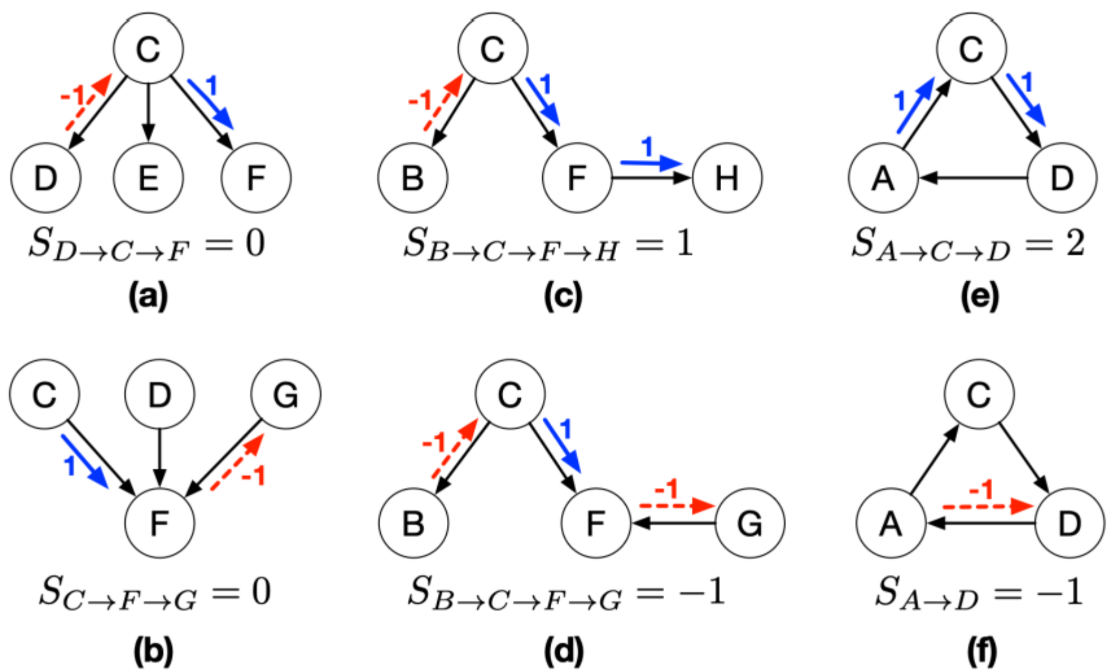
$$P\left(v \mid u, s_{u,v} < 0\right) = \frac{\exp\left(h_v^s \cdot h_u^t\right)}{\sum_{k \in V} \exp\left(h_v^s \cdot h_u^t\right)}$$

$$P\left(v \mid u, s_{u,v} = 0\right) = \frac{\exp\left(h_v^s \cdot h_u^t + h_u^s \cdot h_v^t\right)}{\sum_{k \in V} \exp\left(h_v^s \cdot h_u^t + h_u^s \cdot h_v^t\right)}$$

Direction-Aware User Recommendation Based on Asymmetric Network Embedding (TOIS 2021)

Complex Graph Mining-Directed Graph

- Random-walk-based Methods-infowalk:
  - Capturing hierarchy relations



Direction-Aware User Recommendation Based on Asymmetric Network Embedding (TOIS 2021)

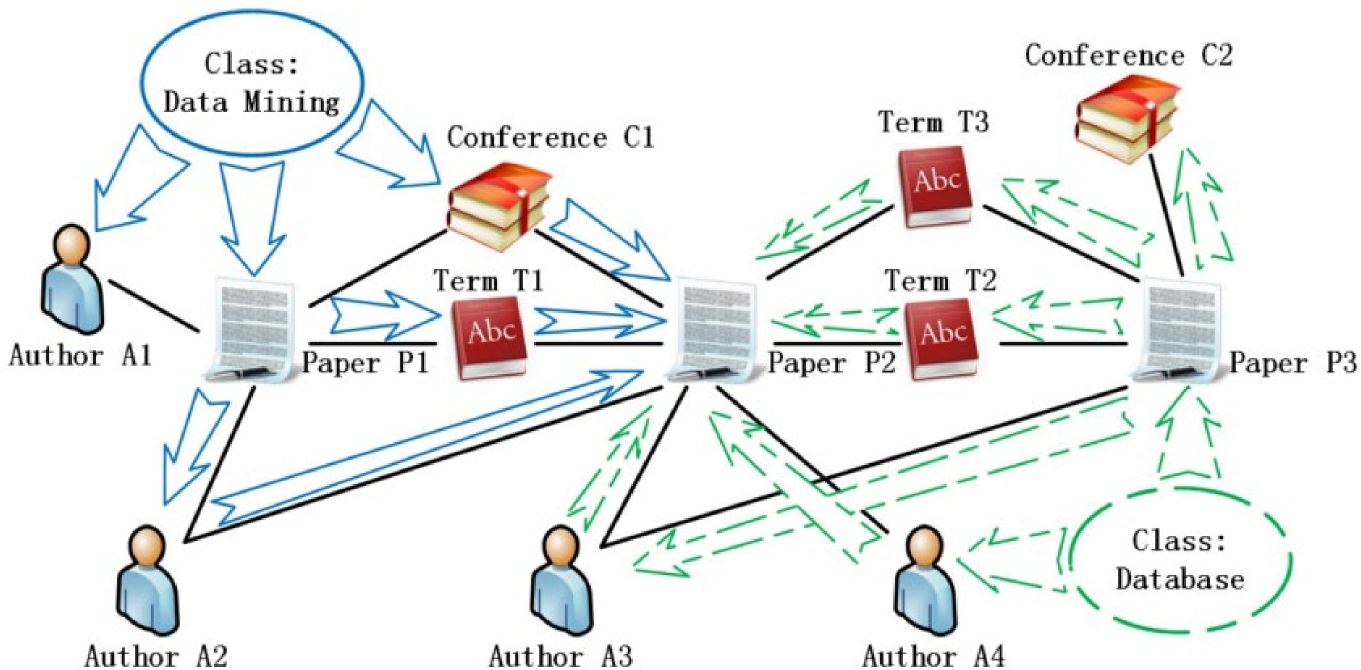
Complex Graph Mining-Directed Graph

- Graph Neural Network for directed graph
  - How to capture the direction signals?

$$
\begin{aligned}
\mathbf{m}_{i,\leftarrow}^{(k)} &= \mathrm{AGG}_{\leftarrow}^{(k)}\left(\left\{\left(\mathbf{x}_j^{(k-1)}, \mathbf{x}_i^{(k-1)}\right) : (j,i) \in E\right\}\right) \\
\mathbf{m}_{i,\rightarrow}^{(k)} &= \mathrm{AGG}_{\rightarrow}^{(k)}\left(\left\{\left(\mathbf{x}_j^{(k-1)}, \mathbf{x}_i^{(k-1)}\right) : (i,j) \in E\right\}\right) \\
\mathbf{x}_i^{(k)} &= \mathrm{COM}^{(k)}\left(\mathbf{x}_i^{(k-1)}, \mathbf{m}_{i,\leftarrow}^{(k)}, \mathbf{m}_{i,\rightarrow}^{(k)}\right).
\end{aligned}
$$

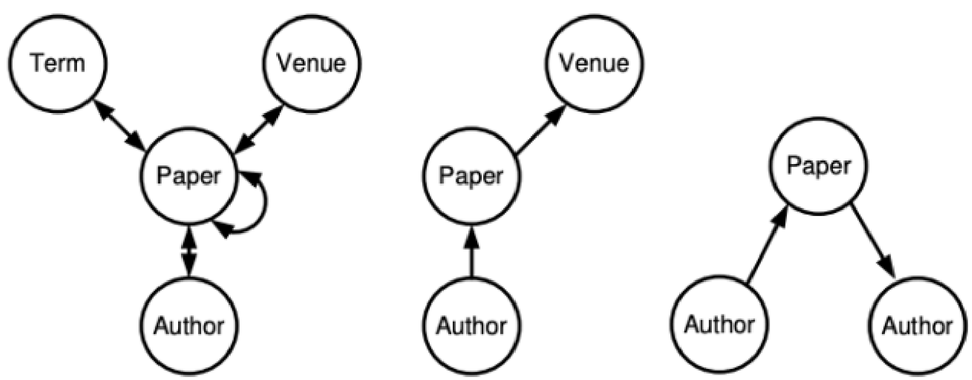
Complex Graph Mining

- Heterogeneous graph
  - Various node types and edge types



Complex Graph Mining-HIN

- Heterogeneous information graph
  - Various node types and edge types
  - Key: **Meta-path:** A path with semantic meaning
    - A sequence consisting of nodes' types (or edges' types)



(a) Network Schema (b) Meta Path: APC (c) Meta Path: APA

Network Schema 和 Meta-path

Complex Graph Mining-HIN

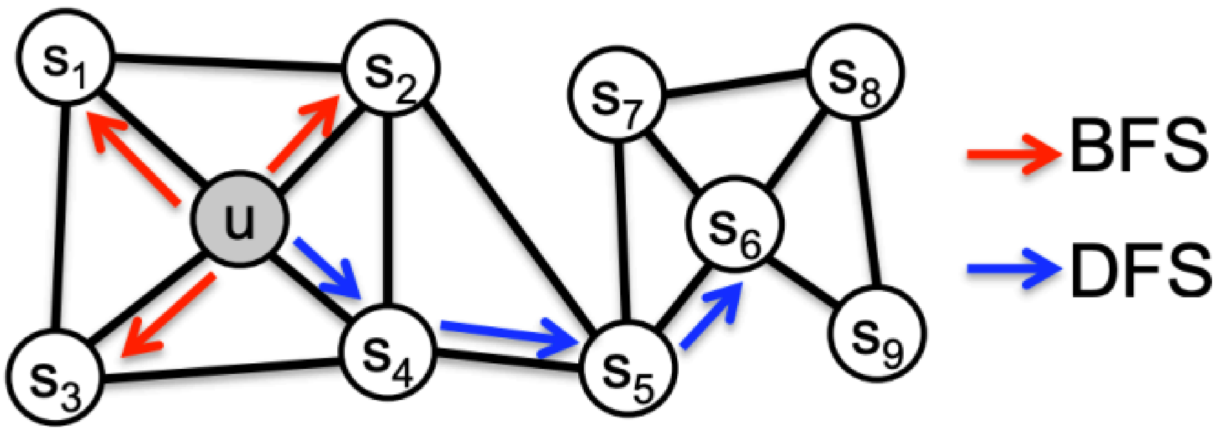
- Heterogeneous information graph
  - Building on the basis of vanilla graph learning methods

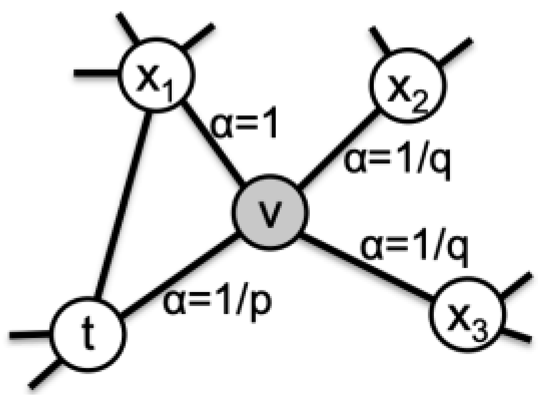
1. Node2Vec $\Rightarrow$ Metapath2Vec
2. SDNE $\Rightarrow$ HIN2Vec
3. GAT $\Rightarrow$ HAN

  - Questions to answer:
    - How to capture the information on the meta-paths?
    - How to inject these information into embeddings?

Complex Graph Mining-HIN

- Heterogeneous information graph-Metapath2vec





Node2vec

Complex Graph Mining-HIN

- Heterogeneous information graph-Metapath2vec
  - Nodes can be connected through different meta-paths.
  - Hetergenous Skip-gram
    - Considering whether a node exists in the context of node according to the t-th type of Meta-path

$$\arg\max_{\theta} \sum_{v \in V} \sum_{t \in T_V} \sum_{c_t \in N_t(v)} \log p\left(c_t \mid v; \theta\right)$$

$$p\left(c_t \mid v; \theta\right) = \frac{e^{X_{c_t} \cdot X_v}}{\sum_{u \in V} e^{X_u \cdot X_v}}$$

metapath2vec: Scalable representation learning for heterogeneous networks (KDD'16)

Complex Graph Mining-HIN

- Heterogeneous information graph-Metapath2vec

  - Sampling strategies of Metapath2vec

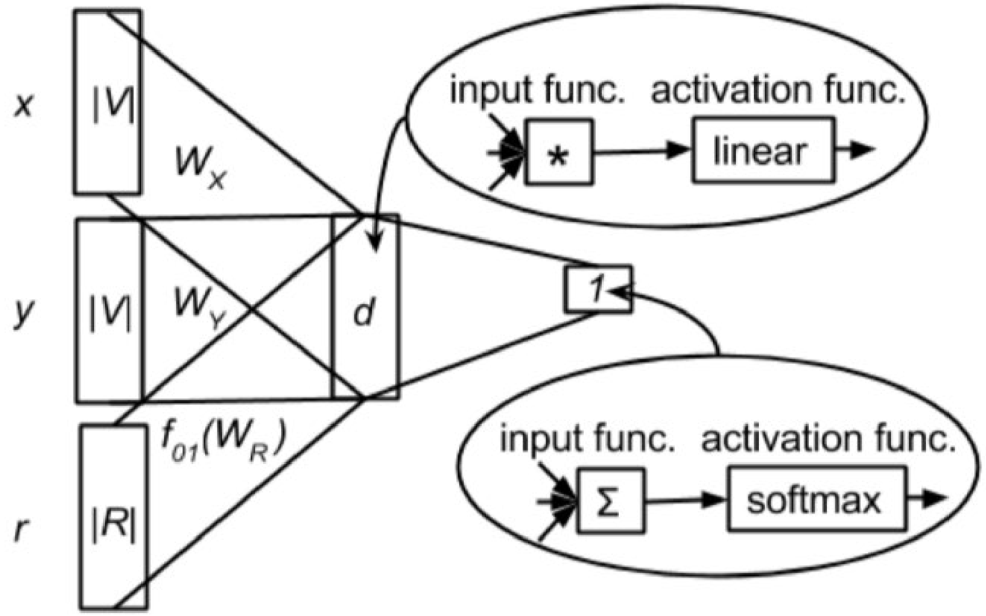
$$p\left(v^{i+1} \mid v_t^i, \mathcal{P}\right) = \begin{cases} \frac{1}{\left|N_{t+1}\left(v_t^i\right)\right|} & \left(v^{i+1}, v_t^i\right) \in E, \phi\left(v^{i+1}\right) = t+1 \\ 0 & \left(v^{i+1}, v_t^i\right) \in E, \phi\left(v^{i+1}\right) \neq t+1 \\ 0 & \left(v^{i+1}, v_t^i\right) \notin E \end{cases}$$

Restricted Random Walk strategy specifically designed for heterogeneous information networks, with other parts being consistent with Node2Vec.

metapath2vec: Scalable representation learning for heterogeneous networks (KDD'16)

Complex Graph Mining-HIN

- Heterogeneous information graph-HIN2vec

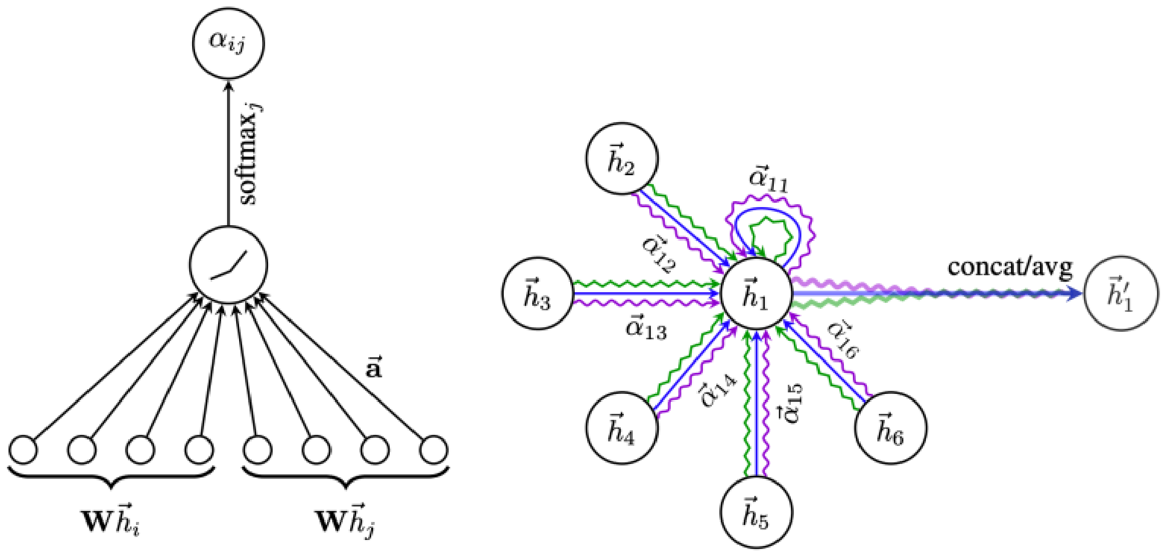


$$\log O_{x,y,r}(x,y,r) = L(x,y,r)\log P(r|x,y) + [1 - L(x,y,r)]\log[1 - P(r|x,y)]$$

HIN2Vec: Explore Meta-paths in Heterogeneous Information Networks for Representation Learning(CIKM2017)

Complex Graph Mining-HIN

- Heterogeneous information graph-Metapath2vec

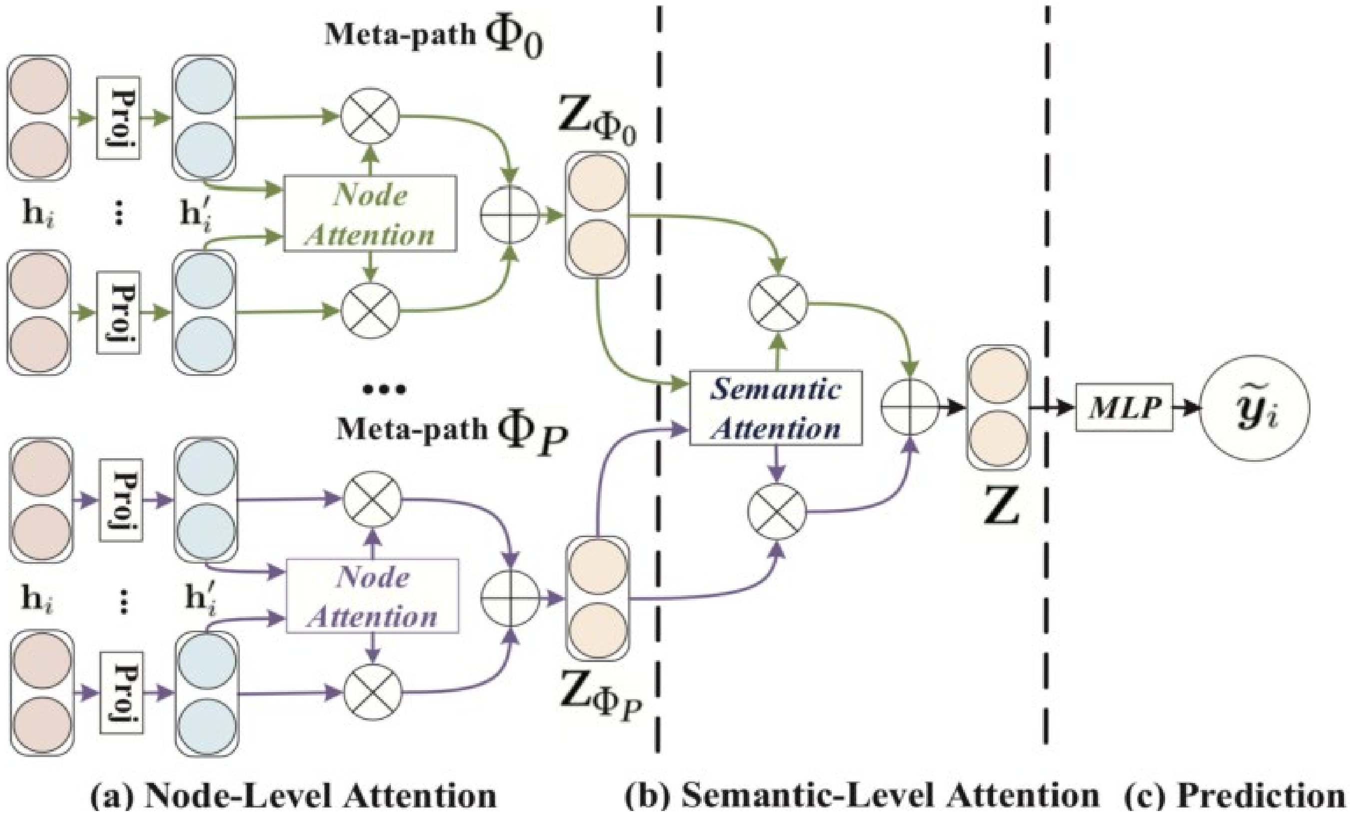


Graph Attention Network

How to leverage attention mechanism in HIN to capture the **diverse importance** of nodes and meta-paths?

Complex Graph Mining-HIN

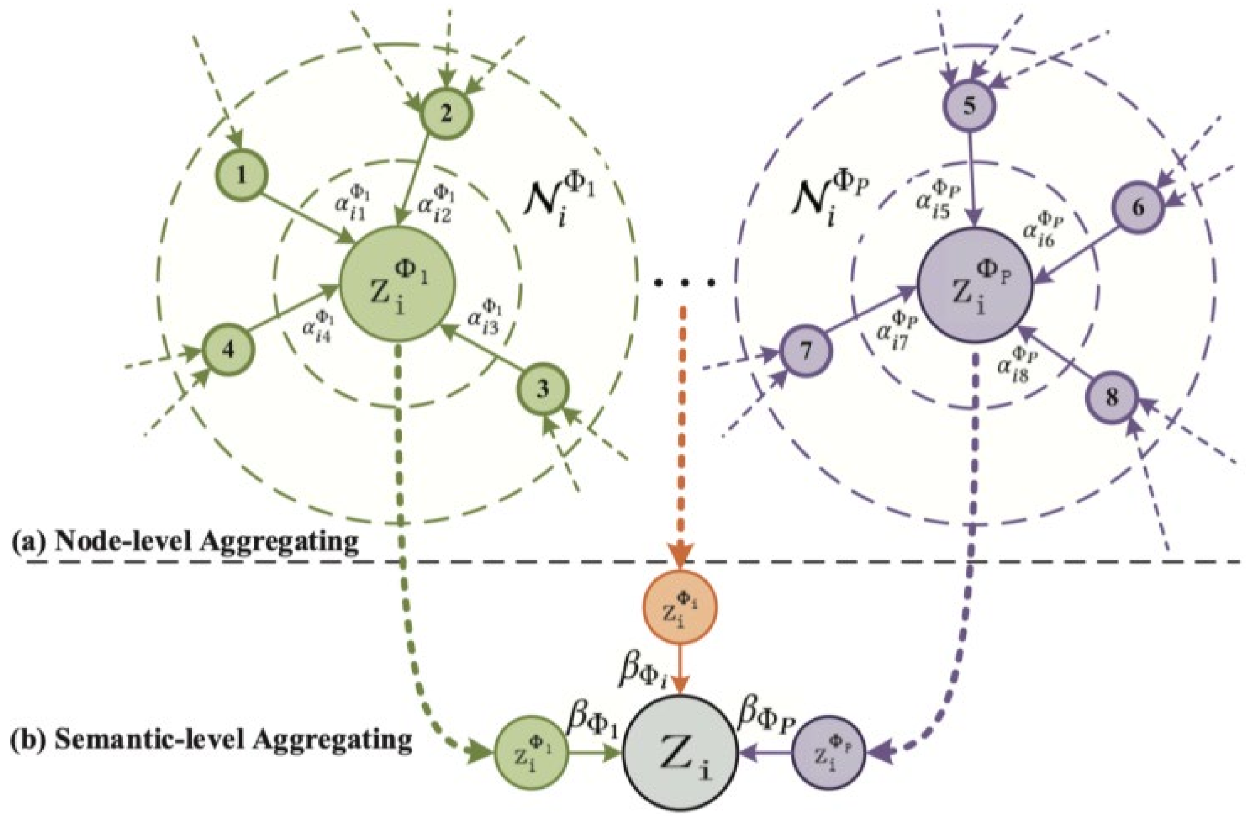
- Heterogeneous information graph-HGAT



HAN 模型架构

Complex Graph Mining-HIN

- Heterogeneous information graph-HGAT



Complex Graph Mining-HIN

- Heterogeneous information graph-HGAT

1. Node-level Attention

$$\alpha_{ij}^{\Phi} = \mathrm{softmax}_j\left(e_{ij}^{\Phi}\right) = \frac{\exp\left(\sigma\left(\mathbf{a}_{\Phi}^{\mathrm{T}} \cdot \left[\mathbf{h}_i' \| \mathbf{h}_j'\right]\right)\right)}{\sum_{k \in \mathcal{N}_i^{\Phi}} \exp\left(\sigma\left(\mathbf{a}_{\Phi}^{\mathrm{T}} \cdot \left[\mathbf{h}_i' \| \mathbf{h}_k'\right]\right)\right)}$$

$$\mathbf{z}_i^{\Phi} = \sigma\left(\sum_{j \in \mathcal{N}_i^{\Phi}} \alpha_{ij}^{\Phi} \cdot \mathbf{h}_j'\right)$$

2. Semantic-level Attention

$$w_{\Phi_p} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathbf{q}^{\mathrm{T}} \cdot \tanh\left(\mathbf{W} \cdot \mathbf{z}_i^{\Phi_p} + \mathbf{b}\right)$$

$$\beta_{\Phi_p} = \frac{\exp\left(w_{\Phi_p}\right)}{\sum_{p=1}^{P} \exp\left(w_{\Phi_p}\right)}$$

$$\mathbf{Z} = \sum^{P} \beta_{\Phi_p} \cdot \mathbf{Z}_{\Phi_p}$$

Complex Graph Mining-Benchmark

- Heterogeneous information graph-Benchmark

Table 3: Node classification benchmark. Vacant positions ("-") mean that the models run out of memory on large graphs.

	DBLP		IMDB		ACM		Freebase	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
RGCN	91.52±0.50	92.07±0.50	58.85±0.26	62.05±0.15	91.55±0.74	91.41±0.75	46.78±0.77	58.33±1.57
HAN	91.67±0.49	92.05±0.62	57.74±0.96	64.63±0.58	90.89±0.43	90.79±0.43	21.31±1.68	54.77±1.40
GTN	93.52±0.55	93.97±0.54	60.47±0.98	65.14±0.45	91.31±0.70	91.20±0.71	-	-
RSHN	93.34±0.58	93.81±0.55	59.85±3.21	64.22±1.03	90.50±1.51	90.32±1.54	-	-
HetGNN	91.76±0.43	92.33±0.41	48.25±0.67	51.16±0.65	85.91±0.25	86.05±0.25	-	-
MAGNN	93.28±0.51	93.76±0.45	56.49±3.20	64.67±1.67	90.88±0.64	90.77±0.65	-	-
HetSANN	78.55±2.42	80.56±1.50	49.47±1.21	57.68±0.44	90.02±0.35	89.91±0.37	-	-
HGT	93.01±0.23	93.49±0.25	63.00±1.19	67.20±0.57	91.12±0.76	91.00±0.76	29.28±2.52	60.51±1.16
GCN	90.84±0.32	91.47±0.34	57.88±1.18	64.82±0.64	92.17±0.24	92.12±0.23	27.84±3.13	60.23±0.92
GAT	93.83±0.27	93.39±0.30	58.94±1.35	64.86±0.43	92.26±0.94	92.19±0.93	40.74±2.58	65.26±0.80
Simple-HGN	**94.01±0.24**	**94.46±0.22**	**63.53±1.36**	**67.36±0.57**	**93.42±0.44**	**93.35±0.45**	**47.72±1.48**	**66.29±0.45**

Seems limited performance gain?

Are we really making much progress? Revisiting, benchmarking, and refining heterogeneous graph neural networks(KDD 2021)

Complex Graph Mining-Benchmark

- Heterogeneous information graph-HGNN

$$\hat{\alpha}_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\boldsymbol{a}^T[\boldsymbol{W}\boldsymbol{h}_i \| \boldsymbol{W}\boldsymbol{h}_j \| W_r \boldsymbol{r}_{\psi(\langle i,j \rangle)}]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\boldsymbol{a}^T[\boldsymbol{W}\boldsymbol{h}_i \| \boldsymbol{W}\boldsymbol{h}_k \| W_r \boldsymbol{r}_{\psi(\langle i,k \rangle)}]\right)\right)},$$

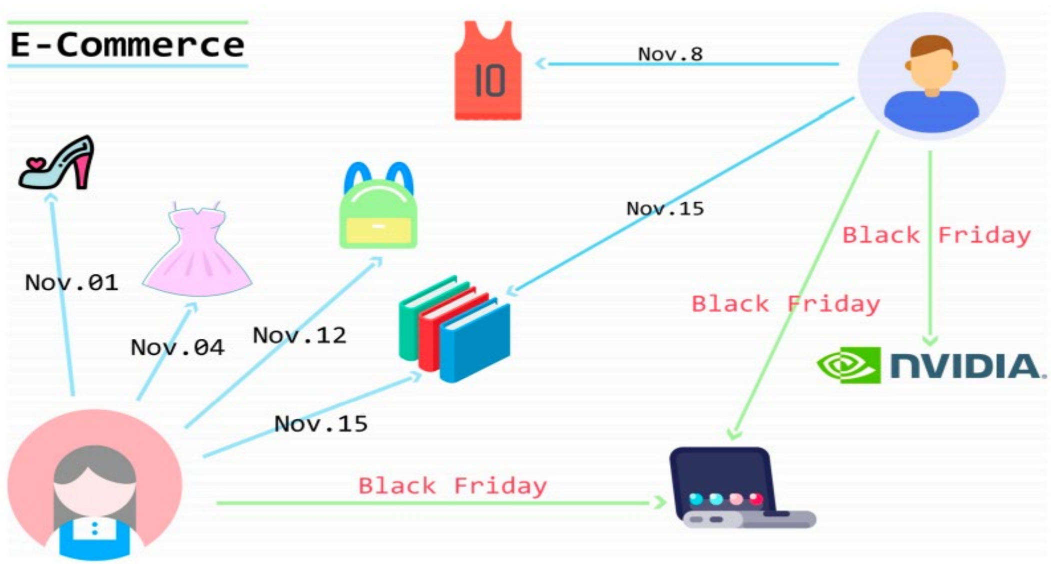
$$\boldsymbol{h}_i^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(l)} \boldsymbol{W}^{(l)} \boldsymbol{h}_j^{(l-1)} + \boldsymbol{W}_{\text{res}}^{(l)} \boldsymbol{h}_i^{(l-1)}\right).$$

$$\alpha_{ij}^{(l)} = (1-\beta)\hat{\alpha}_{ij}^{(l)} + \beta \alpha_{ij}^{(l-1)},$$

Are we really making much progress? Revisiting, benchmarking, and refining heterogeneous graph neural networks(KDD 2021)

Complex Graph Mining

- Dynamic(Temporal) Graph
  - Networks in the real world are not static, but dynamic
  - node and edges would evolve over time

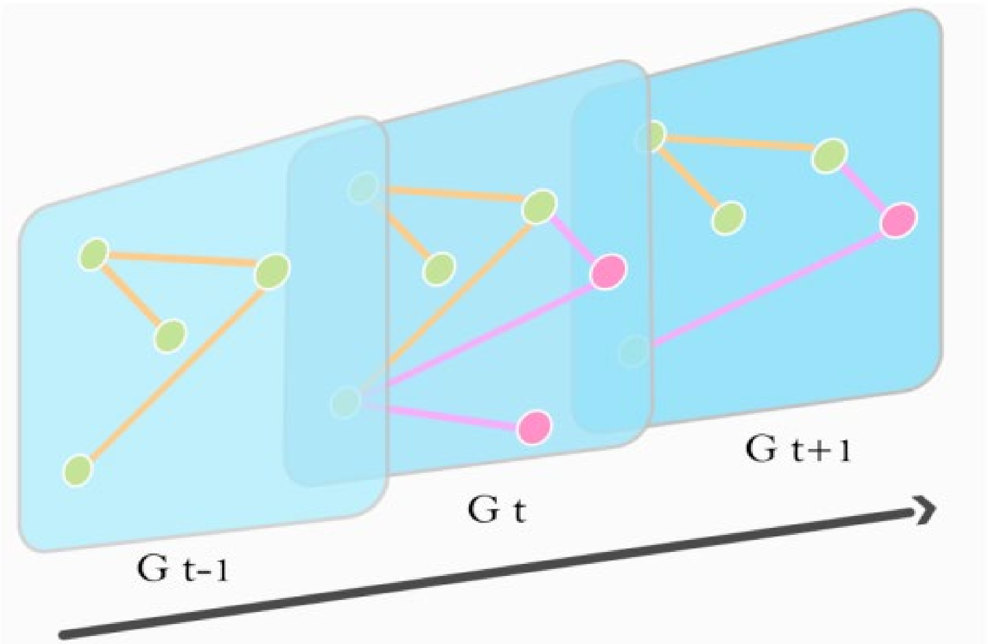


Complex Graph Mining

- Challenges of dynamic graph learning
  - How to formally define dynamic graph?
  - Edges and nodes evolve over time, and **neighborhood aggregation** is subject to temporal constraints
  - How to encode the temporal information into embedding

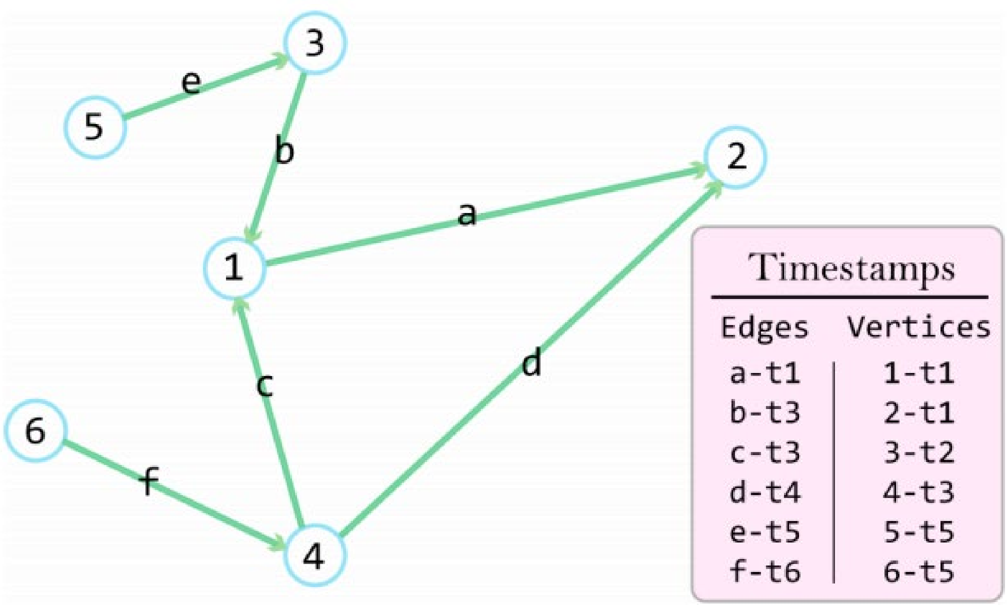
Complex Graph Mining

- Dynamic(Temporal) Graph-definition
  - Discrete dynamic graph model
    - A chronological series of static graph snapshots



Complex Graph Mining

- Dynamic(Temporal) Graph-definition
  - Continual dynamic graph model
    - A series of edge event streams



Complex Graph Mining

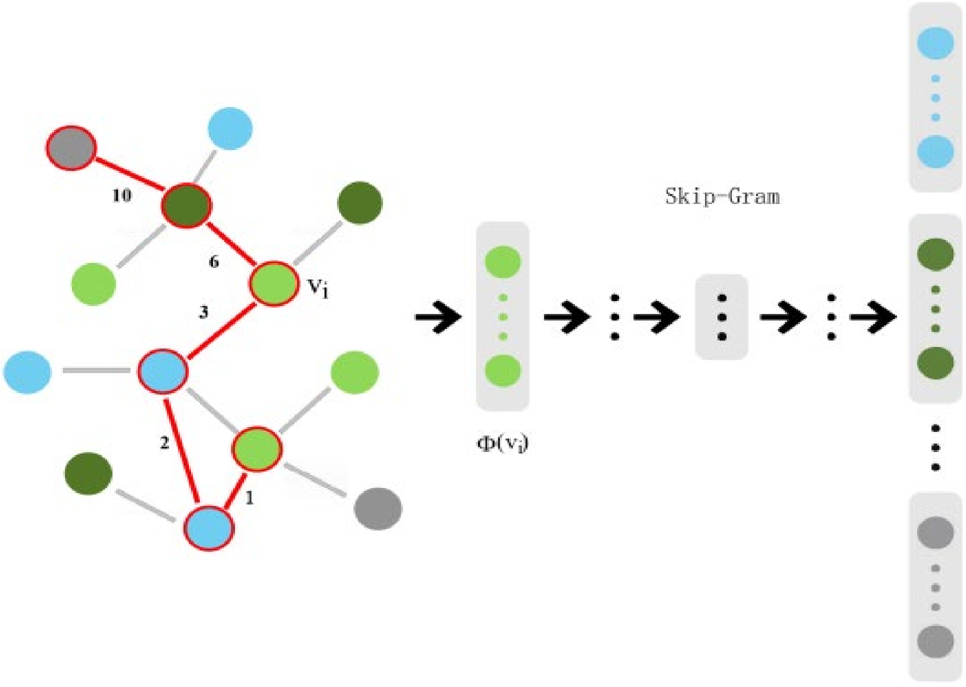
- Dynamic(Temporal) Graph-definition
  - Continual VS. Discrete
    - Continual is more general
    - Continual dynamic graph incurs less memory cost
    - Suitable for scenarios with variable length time intervals
    - Continual drawing increasing attention

Complex Graph Mining-Dynamic Graph

- Build on the foundation of vanilla graph model (借鉴)
  - Random-walker-based methods
  - Auto-encoder-based methods
  - Graph neural network-based methods

Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-CTDNE
  - each edge has a corresponding timestamp
  - the nodes in the random walk are connected by a series of edges with incrementing timestamps



Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-CTDNE
  - Generating a sequence of temporal edges

$$W = ((w_m, t_m), (w_{m-1}, t_{m-1}), \ldots, (w_0, t_0))$$
$$\text{s.t.} \quad t_m > t_{m-1} > \ldots > t_0, (w_i, w_{i-1}, t_{i-1}) \in \mathcal{G}$$

The temporal walk starts from the node with a later timestamp, walks along the temporal edges, and arrives at the node with an earlier timestamp.

Continuous-Time Dynamic Network Embeddings (WWW 2018)

Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-CTDNE
  - Strategies of starting point
    - Unbiased: uniform

$$Pr(e) = \frac{1}{|\mathcal{G}|}$$

    - Biased: Edges with later timestamps have a higher probability of being sampled

$$Pr(e) = \frac{\exp(t - t_{min})}{\sum_{e' \in \mathcal{G}} \exp(t' - t_{min})}$$

Using temporal information!

Continuous-Time Dynamic Network Embeddings (WWW 2018)

Complex Graph Mining-Dynamic Graph

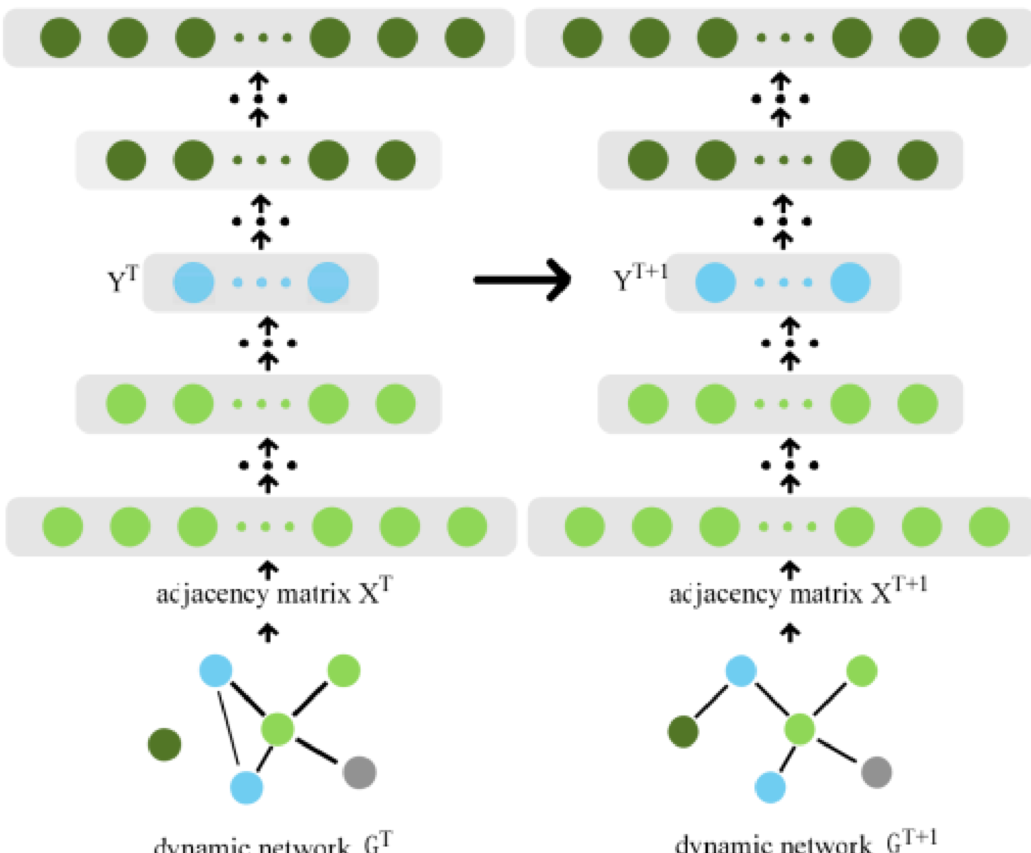
- Dynamic(Temporal) Graph-CTDNE
  - Strategies of learning
    - employing the concept of Skip-Gram to maximize the co-occurrence probability of node embeddings within the window

$$Pr(W = |f(v_i)) = \prod_{v_{i+k} \in W} Pr(v_{i+k}|f(v_i))$$

Continuous-Time Dynamic Network Embeddings (WWW 2018)

Complex Graph Mining-Dynamic Graph

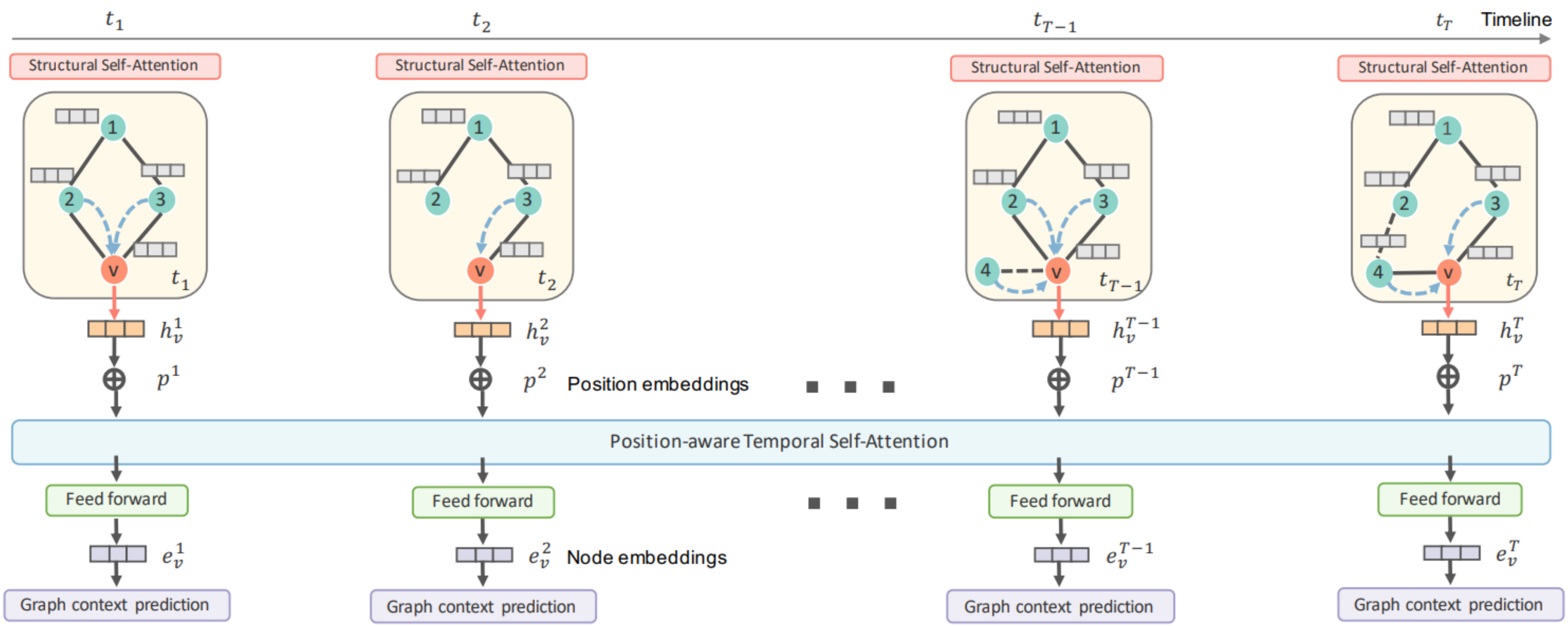
- Dynamic(Temporal) Graph-Auto-encoder



- ❑ In the discrete model
- ❑ the weight parameters from the previous snapshot are used to initialize the network for the next snapshot

Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-Dysat



Running static graph methods for each snapshot, and then performing sequence model to capture temporal relations

Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-Dysat

$$Z_v = \boldsymbol{\beta_v}(\mathbf{X}_v \mathbf{W}_v), \qquad \beta_v^{ij} = \frac{\exp(e_v^{ij})}{\sum_{k=1}^{T} \exp(e_v^{ik})},$$

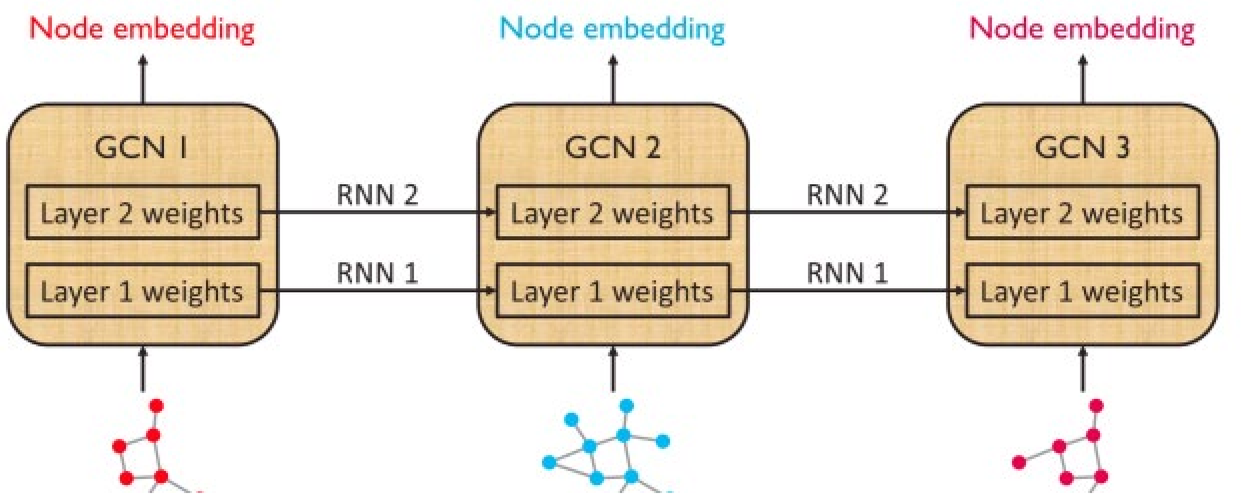
$$e_v^{ij} = \left(\frac{((\mathbf{X}_v \mathbf{W}_q)(\mathbf{X}_v \mathbf{W}_k)^T)_{ij}}{\sqrt{F'}} + M_{ij} \right)$$

$$M_{ij} = \begin{cases} 0, & i \leq j \\ -\infty, & \text{otherwise} \end{cases}$$

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks (WSDM’20)

Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-evolveGCN

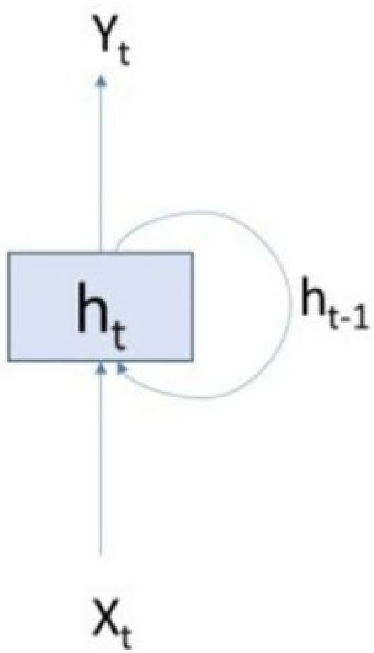


EvolveGCN adopts parameter-level: using a RNN to capture the dynamic patterns of the parameters of the GNN rather than representation

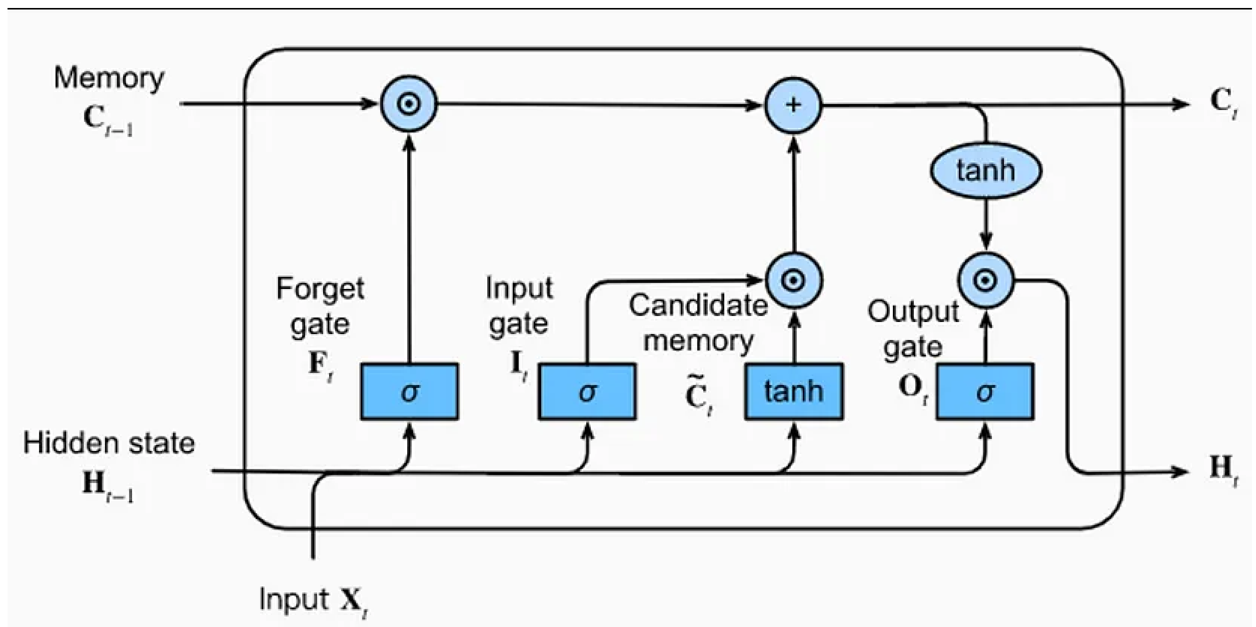
Parameters are relatively more stable.

Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-evolveGCN
  - LSTM



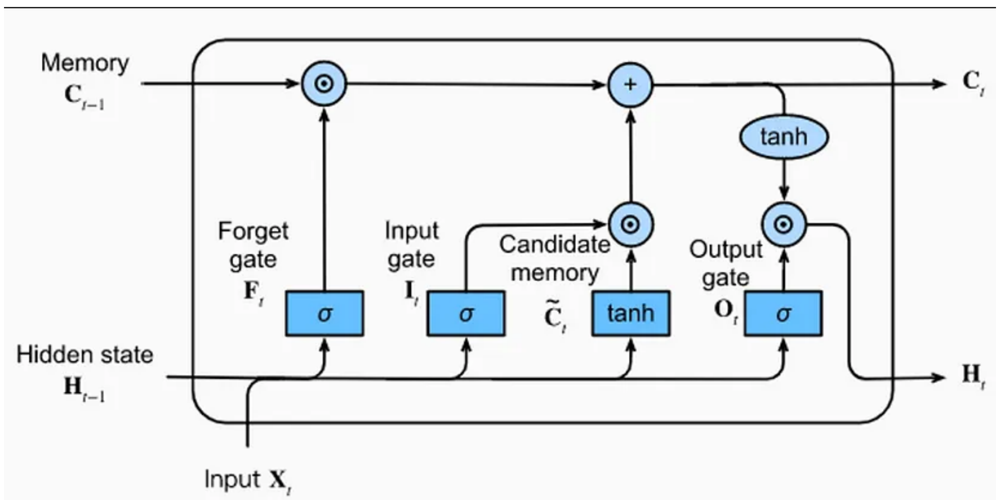
RNN



LSTM

Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-evolveGCN
  - LSTM



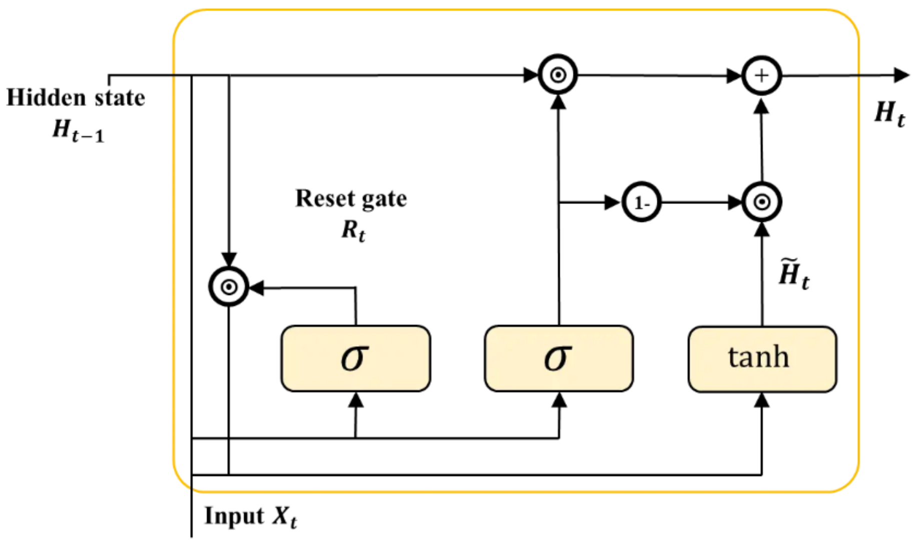
$$
\begin{aligned}
f_t &= \sigma(W_{hf}x_t + W_{hf}h_{t-1} + b_f) \\
i_t &= \sigma(W_{hi}x_t + W_{hi}h_{t-1} + b_i) \\
\tilde{C}_t &= \tanh(W_{hc}x_t + W_{hc}h_{t-1} + b_c) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
h_t &= \tanh(C_t) \\
o_t &= \sigma(W_{ho}x_t + W_{ho}h_{t-1} + b_o) \\
h_t &= o_t * \tanh(C_t)
\end{aligned}
$$

f:记忆门，要保留多少信息；
i: 输入门，输入信号要保留多少；
C： 记忆模块，之前信息的叠加；
h: 当前的状态，隐层表征

基本思想：搞个记忆单元，来存储长期的信息

Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-evolveGCN
  - GRU: simpler and efficient



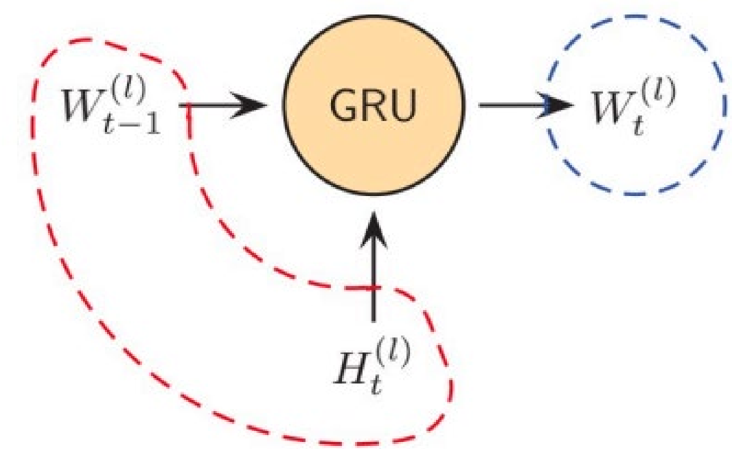
- Update gate:
  $z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$
- Reset gate:
  $r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$
- Candidate hidden state:
  $\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t] + b)$
- New hidden state:
  $h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$

主要区别：合并了c和f,没有了输入门，没有了输出门

Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-evolveGCN
  - EvolveGCN-H: Parameters as hidden state

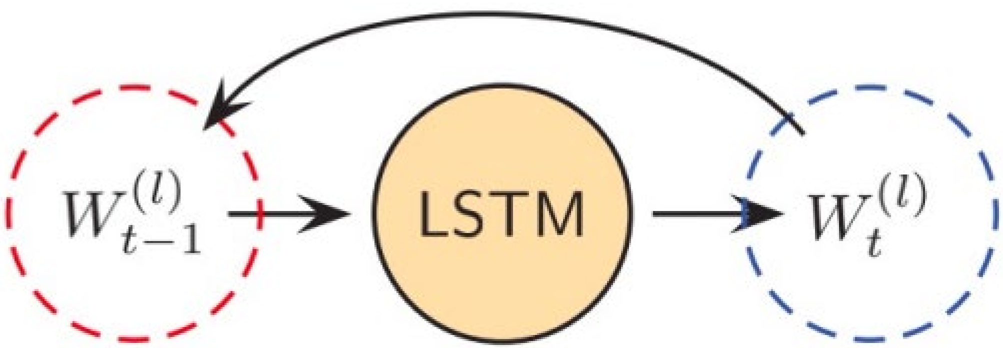
$$\overbrace{\underbrace{W_t^l}_{\text{hidden state}}}^{\text{GCN weights}} = \mathbf{GRU}(\overbrace{\underbrace{H_t^l}_{\text{input}}}^{\text{node embeddings}} + \overbrace{\underbrace{W_{t-1}^l}_{\text{hidden state}}}^{\text{GCN weights}})$$



Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-evolveGCN
  - EvolveGCN-O: Parameters as input and output

$$\overbrace{\underbrace{W_t^l}_{\text{output}}}^{\text{GCN weights}} = \mathbf{LSTM}(\overbrace{\underbrace{W_{t-1}^l}_{\text{input}}}^{\text{GCN weights}})$$



Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-evolveGCN
  - EvolveGCN-O Vs. EvolveGCN-H

$$\text{EvolveGCN-H:} \quad \overbrace{\underbrace{W_t^l}_{\text{hidden state}}}^{\text{GCN weights}} = \mathbf{GRU}\Big(\overbrace{\underbrace{H_t^l}_{\text{input}}}^{\text{node embeddings}} + \overbrace{\underbrace{W_{t-1}^l}_{\text{hidden state}}}^{\text{GCN weights}} \Big)$$

$$\text{EvolveGCN-O:} \quad \overbrace{\underbrace{W_t^l}_{\text{output}}}^{\text{GCN weights}} = \mathbf{LSTM}\Big(\overbrace{\underbrace{W_{t-1}^l}_{\text{input}}}^{\text{GCN weights}} \Big)$$

  - When node attributes are abundant, the H scheme is more effective, as its RNN includes additional node representation inputs.
  - When the structure of the network is more important, the O scheme focuses more on structural changes and could be better.

Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-TGAT for **continual DG**
  - TGAT extends the message-passing to dynamic graphs
  - sampling and aggregating the representations of historical neighbors layer by layer

$$\mathbf{Z}(t) = [\tilde{h}_0^{(l-1)}(t)||\Phi(0), \tilde{h}_1^{(l-1)}(t)||\Phi(t-t_1), ..., \tilde{h}_N^{(l-1)(t)}||\Phi(t-t_N)]$$

time representation function, mapping time differences into vectors.

Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-TGAT for **continual DG**

1) Employs an attention mechanism, obtaining the query, key, and value:

$$\mathbf{q}(t) = [\mathbf{Z}(t)]_0\mathbf{W}_Q, \mathbf{K}(t) = [\mathbf{Z}(t)]_{1:N}\mathbf{W}_Q, \mathbf{V}(t) = [\mathbf{Z}(t)]_{1:N}\mathbf{W}_V$$

2) After the local attention layer:

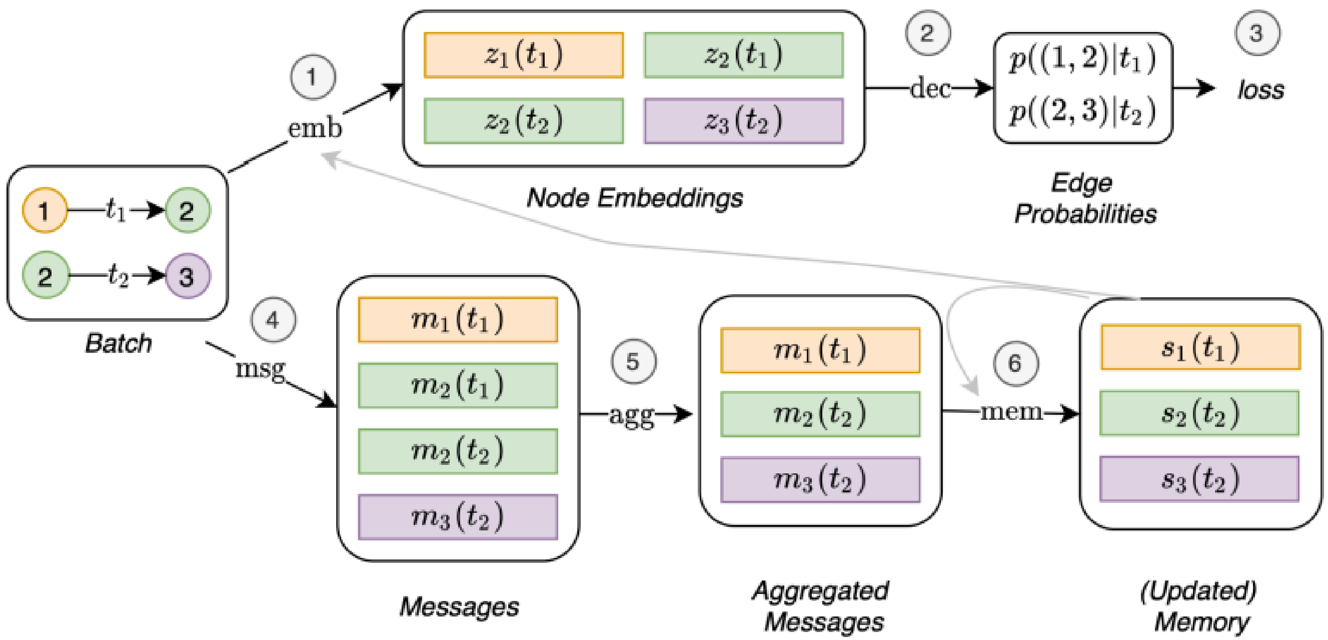
$$\mathbf{h}(t) = \mathrm{Attn}(\mathbf{q}(t), \mathbf{K}(t), \mathbf{V}(t)) \in \mathbb{R}^{d_h},$$

3) Generate the reprensetation:

$$\tilde{\mathbf{h}}_0^{(l)}(t) = \mathrm{FFN}(\mathbf{h}(t)||\mathbf{x}_0)$$

Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-TGN for **continual DG**
  - Improving TGAT with a memory mechanism



Temporal Graph Networks For Deep Learning On Dynamic Graphs (ICML 2020 Workshop)

Complex Graph Mining-Dynamic Graph

- Dynamic(Temporal) Graph-TGN for **continual DG**
  - Generating messages when an edge event occurs

$$\mathbf{m}_i(t) = \mathrm{msg}(\mathbf{s}_i(t^-), \mathbf{s}_j(t^-), \Delta t, \mathbf{e}_{ij}(t))$$
$$\mathbf{m}_j(t) = \mathrm{msg}(\mathbf{s}_j(t^-), \mathbf{s}_i(t^-), \Delta t, \mathbf{e}_{ij}(t))$$

  - Updating nodes memories

$$\mathbf{s}_i(t) = \mathrm{mem}(\mathbf{m}_i(t), \mathbf{s}_i(t^-))$$

Complex Graph Mining-Dynamic Graph

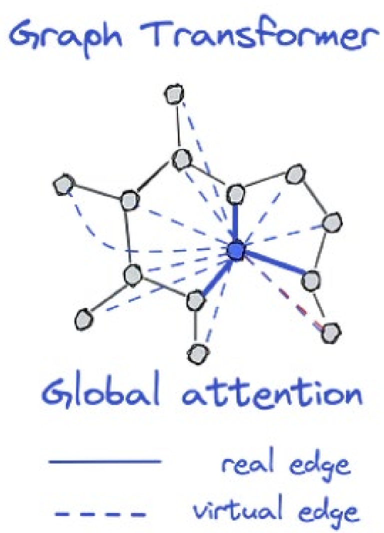
- Dynamic(Temporal) Graph-TGN for **continual DG**
  - Updating nodes' embeddings with neighbors and memories

$$\mathbf{z}_i(t) = \sum_{j \in \eta_i^K(t)} h(\mathbf{s}_i(t), \mathbf{s}_j(t), \mathbf{s}_{ij}, \mathbf{x}_i, \mathbf{x}_j)$$

  - A general framework subsumes existing strategies

Complex Graph Mining-Directions

- Better models (借鉴的思路)
  - Building on the shoulder of vanilla methods
  - Transformers for complex graph
    - Position encoding capturing spatio-temporal patterns



Complex Graph Mining-Directions

- New tasks（新问题)
  - Building on the shoulder of vanilla graph
  - Emphasizing the importance and challenges of the problems
    - Robustness (attacks, noisy, OOD, etc)
    - Fairness
    - Privacy
    - Anomy detection
    - Incremental learning

    ….

Complex Graph Mining-Directions

- Theoretical analyses (分析型)
  - Analyzing the nature of the complex graph and existing methods
    - Why existing methods work? Based on which assumptions?
    - Expressive power of existing methods on complex graph
    - Heterophily complex graph?

Thank You!