Dynamics in GNNs

Guest Lecture at CS6804 – Machine Learning on Graphs

Dongqi Fu
Ph.D. Candidate
Department of Computer Science
University of Illinois, Urbana-Champaign
dongqif2@illinois.edu
https://dongqifu.github.io/

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

VIRGINIA TECH

Contents

Basics of graph neural networks (GNNs)

- According to [1], the general formula of GNNs can be expressed as

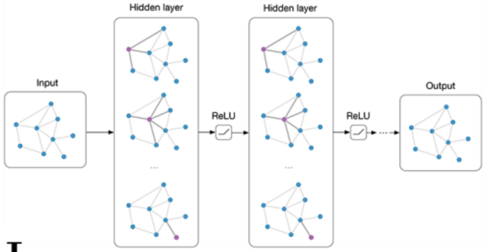
message-passing: information aggregation among hidden representation vectors of neighbors

$\boldsymbol{h}_v^{(k)}$: is the hidden representation of node v at the k-th layer

$$\mathbf{a}_v^{(k)} = AGGREGATE^{(k)}(\{\mathbf{h}_u^{(k-1)} : u \in \mathcal{N}(v)\}), \ \mathbf{h}_v^{(k)} = COMBINE^{(k)}(\mathbf{h}_v^{(k-1)}, \mathbf{a}_v^{(k)})$$

- For example, the graph convolutional neural network (GCN) [2] can be instanced as

$$\mathbf{h}_v^{(k)} = ReLU(\mathbf{W}^{(k-1)} \cdot MEAN\{\mathbf{h}_u^{(k-1)}, \forall u \in \mathcal{N}(v) \cup \{v\}\})$$



with the original formula as

$$\mathbf{H}^{(k)} = ReLU(\hat{\mathbf{A}}\mathbf{H}^{(k-1)}\mathbf{W}^{(k-1)}) \qquad \hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}} \qquad \tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$$

[1] Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka: How Powerful are Graph Neural Networks? ICLR 2019
[2] Thomas N. Kipf, Max Welling: Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2017

GNNs have broad application domains

Computer Vision [1]

Natural Language Processing [2]

Recommender Systems [3]

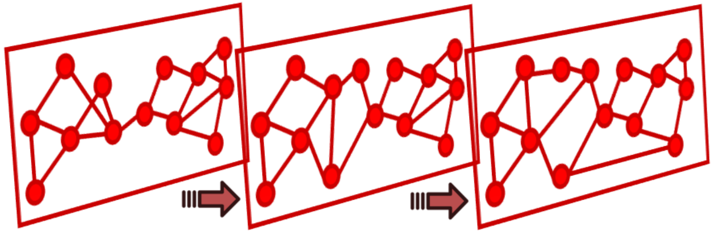
Drug Discovery [4]

image source: https://realpython.com/

[1] Chen et al.: A Survey on Graph Neural Networks and Graph Transformers in Computer Vision: A Task-Oriented Perspective. CoRR 2022
[2] Wu et al.: Graph Neural Networks for Natural Language Processing: A Survey. CoRR 2021
[3] Wang et al.: Graph Learning based Recommender Systems: A Review. IJCAI 2021
[4] Gaudelet et al.: Utilizing Graph Machine Learning within Drug Discovery and Development. Briefings in Bioinformatics 2021

What are natural dynamics?

- Natural dynamics in graphs [1]
  - Input graphs have the time-evolving components, e.g.,
    - Topology Structures
    - Node-level, edge-level, and (sub)graph-level features, etc.
  - Continuous Time
    - $\mathcal{G} = \{A, e = (i, j, t, +/-)\}$
  - Discrete Time
    - $\mathcal{G} = \{A^{(1)}, A^{(1)}, \ldots, A^{(T)}\}$

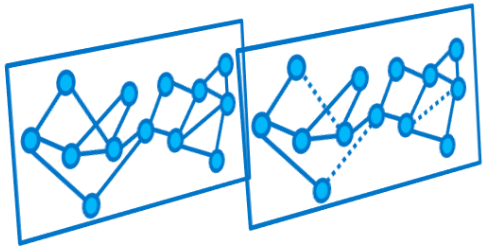


Evolving Graph Structures (Discrete Time Representation)

[1] Dongqi Fu and Jingrui He: Natural and Artificial Dynamics in Graphs: Concept, Progress, and Future. Frontiers in Big Data 2022

What are artificial dynamics?

- Artificial dynamics in graphs [1], researchers and practitioners
  - **change** (e.g., filter, mask, drop, or augment) **the existing** or
  - **construct the non-existing** graph-related elements, e.g.,
    - graph topology
    - node/graph attributes
    - GNN gradients, etc.
  - to realize the certain **performance upgrade**, e.g.,
    - decision accuracy
    - computation efficiency
    - model explanation, etc.



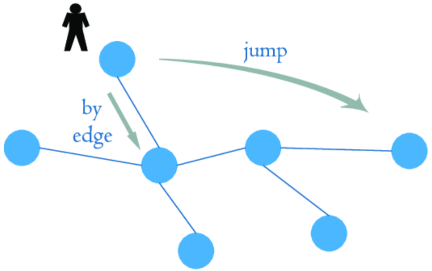
Random Edge Dropping ($A \rightarrow \bar{A}$)

[1] Dongqi Fu and Jingrui He: Natural and Artificial Dynamics in Graphs: Concept, Progress, and Future. Frontiers in Big Data 2022

What are artificial dynamics?

- Artificial dynamics in graphs [1], researchers and practitioners
  - **change** (e.g., filter, mask, drop, or augment) **the existing** or
  - **construct the non-existing** graph-related elements
  - to realize the certain **performance upgrade**

- In 2003, “artificial jump” [2] is proposed to adjust the graph topology for PageRank realizing the personal ranking function on graphs



[1] Dongqi Fu and Jingrui He: Natural and Artificial Dynamics in Graphs: Concept, Progress, and Future. Frontiers in Big Data 2022
[2] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, Gene H. Golub: Extrapolation methods for accelerating PageRank computations. WWW 2003

Relation between natural and artificial dynamics?

- For natural dynamics,
  - The input graph itself is a sequence of observations based on time
  - E.g., daily world wide web like Facebook, Twitter, etc.

- For artificial dynamics,
  - Researchers and practitioners deliberately modify the components for different interests
  - E.g., imperfect or redundant connections, missing features, etc.

- Can they be combined, i.e., **natural + artificial dynamics**?
  - **Yes**, when the input graph is temporal, and the modification is necessary

How natural dynamics contribute GNNs?

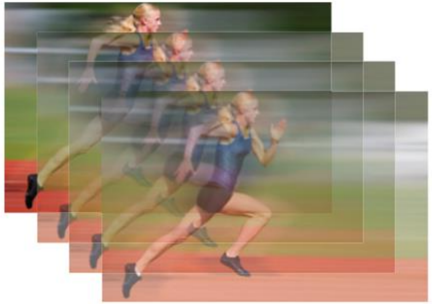
- Considering natural dynamics can help graph machine learning models to capture the temporal correlations among features [1]

- Running?
- Dancing?
- Or just the static model for photography?

[1] Kazemi et al.: Representation Learning for Dynamic Graphs: A Survey. JMLR (2020)

How natural dynamics contribute GNNs?

- Considering natural dynamics can help graph machine learning models to capture the temporal correlations among features [1]

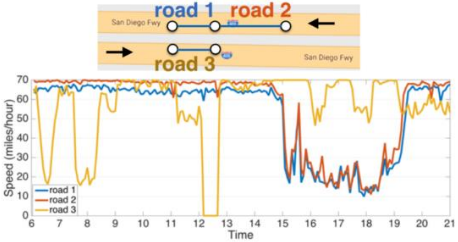


- Running? ✓
- Dancing?
- Or just the static model for photography?

[1] Kazemi et al.: Representation Learning for Dynamic Graphs: A Survey. JMLR (2020)

How natural dynamics contribute GNNs?

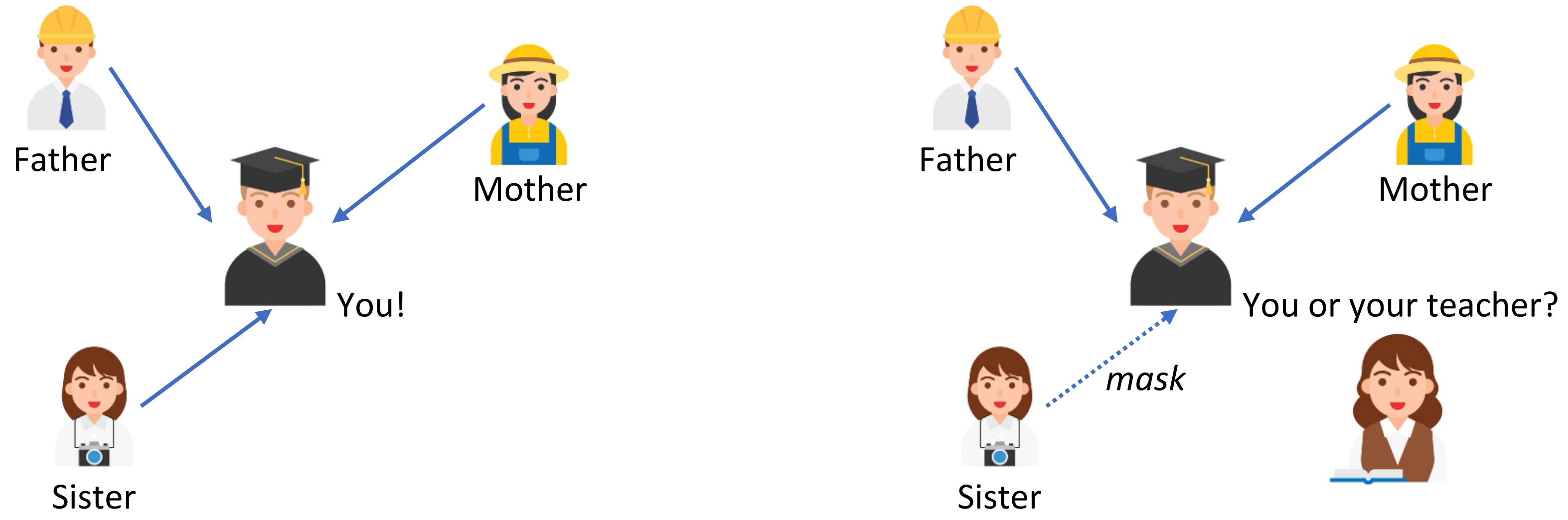
- Considering natural dynamics can help graph machine learning models to capture the temporal correlations among features [1]
  - Motion Recognition [2]
  - Time-Series Forecasting [3]
  - Pandemic Classification [4]
  - Social Network Analysis [5]
  - Many more …



[1] Kazemi et al.: Representation Learning for Dynamic Graphs: A Survey. JMLR (2020)
[2] Yan et al.: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018
[3] Li et al.: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018
[4] Tsiotas et al.: The Effect of Anti-COVID-19 Policies on the Evolution of the Disease: A Complex Network Analysis of the Successful Case of Greece. Physics (2020)
[5] Aggarwal et al.: Evolutionary Network Analysis: A Survey. ACM Comput. Surv. (2014)

How artificial dynamics contribute GNNs?

- Considering artificial dynamics can boost graph machine learning performance [1]

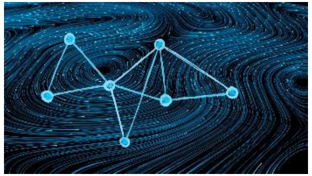


Social Network Anonymization

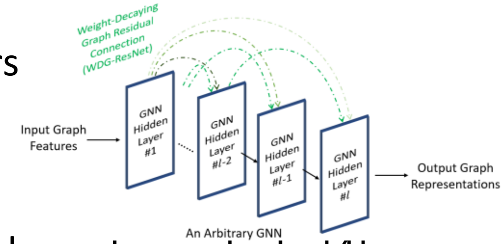
[1] Dongqi Fu and Jingrui He: Natural and Artificial Dynamics in Graphs: Concept, Progress, and Future. Frontiers in Big Data 2022

How artificial dynamics contribute GNNs?

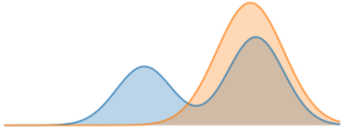
- Considering artificial dynamics can boost graph machine learning performance [1]
  - Privacy-Preserving [2]
    - **Permute the GNN gradients** under the differential privacy constraint



  - Decision Accuracy [3]
    - **Add dependency constraints on weight matrices** of GNN layers



  - Domain Adaption [4]
    - **Graph promoting** for large-scale pre-trained graph models on downstream tasks [4]

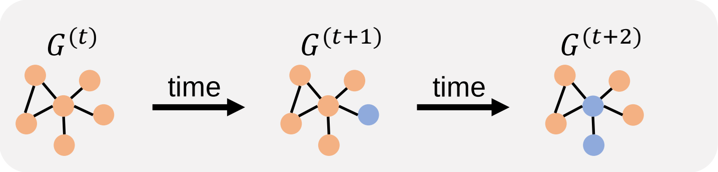


  - Many more …

[1] Dongqi Fu and Jingrui He: Natural and Artificial Dynamics in Graphs: Concept, Progress, and Future. Frontiers in Big Data 2022
[2] Yang et al: Secure Deep Graph Generation with Link Differential Privacy. IJCAI 2021
[3] Zheng et al.: Deeper-GXX: Deepening Arbitrary GNNs. CoRR 2022
[4] Sun et al.:GPPT: Graph Pre-training and Prompt Tuning to Generalize Graph Neural Networks. KDD 2022

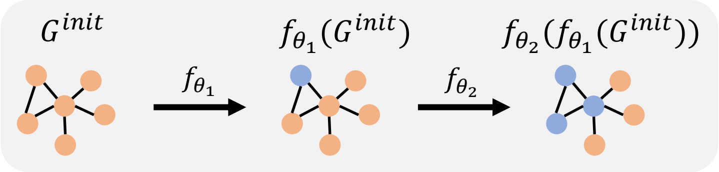
Scope of this tutorial

- Natural dynamics in GNNs
  - We focus on **time-evolving** graph **structures** and node **features**



Naturally Evolving Features

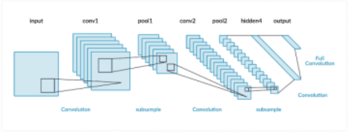
- Artificial dynamics in GNNs
  - We focus the **augmentation strategies** on graph **structures** and node **features**



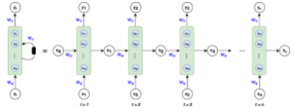
Artificially Evolving Features

Today

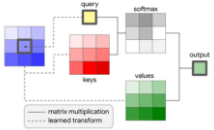
- Covered works for natural dynamics in GNNs
  - Temporal GNNs with Convolutional Operations



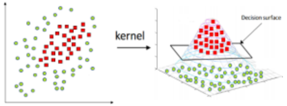
  - Temporal GNNs with Recurrent Units



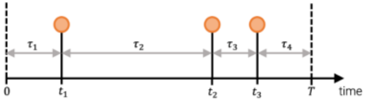
  - Temporal GNNs with Time Attention



  - Temporal GNNs with Time Kernel



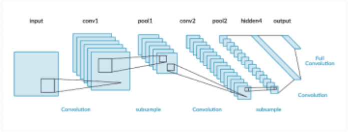
  - Temporal GNNs with Temporal Point Process



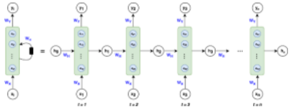
Contents

Roadmap for natural dynamics

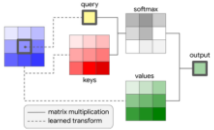
- Covered works for natural dynamics in GNNs
  - Temporal GNNs with Convolutional Operations



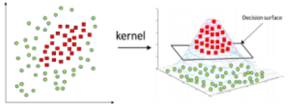
  - Temporal GNNs with Recurrent Units



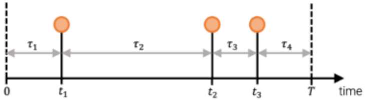
  - Temporal GNNs with Time Attention



  - Temporal GNNs with Time Kernel



  - Temporal GNNs with Temporal Point Process



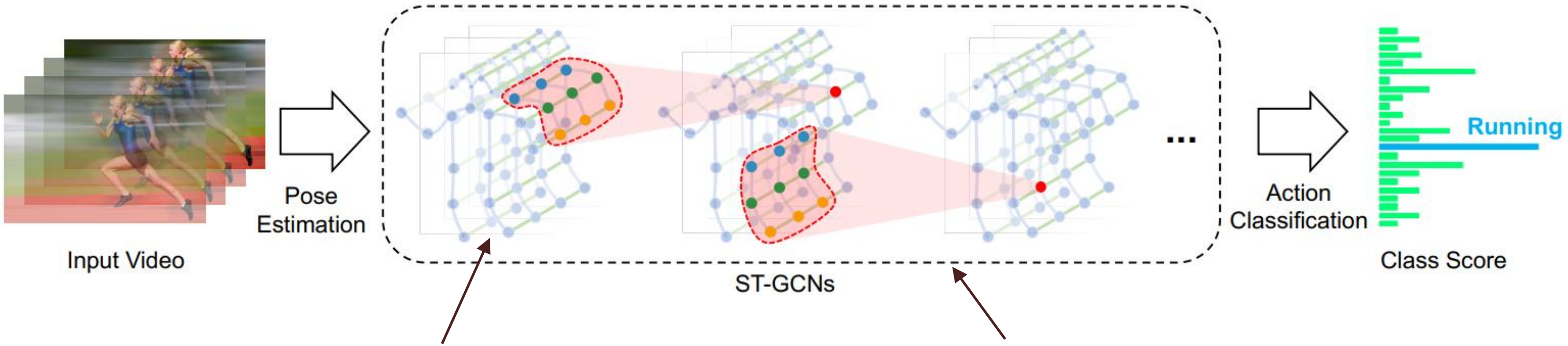
Spatial Temporal Graph Convolutional Networks (ST-GCN) [1]

- **Task:** graph-level representation learning
- **Natural dynamic:** evolving structures and node features w.r.t time
- **Goal:** graph classification

[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

Spatial Temporal Graph Convolutional Networks (ST-GCN) [1]

- Problem Setting
  - Skeleton-based Action Reconstruction
    - or temporal graph classification in the graph research community



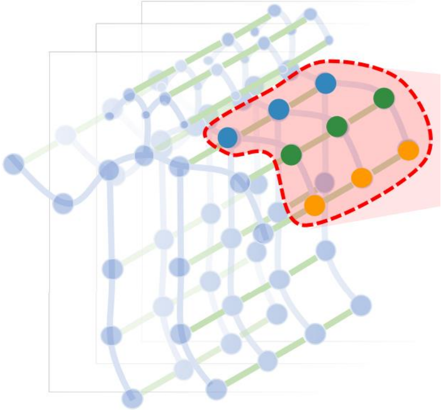
A Skeleton Sequence
(or a Spatial-Temporal Graph)

Going though the Spatial Graph Convolutional Network
to obtain its high-level feature map

[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

Spatial Temporal Graph Convolutional Networks (ST-GCN) [1]

- Graph Modeling
  - $G = (V, E)$ on a skeleton sequence with N joints and T timestamps featuring both intra-body and inter-frame connections
  - $V = \{v_{ti} \mid t = 1, \ldots, T, i = 1, \ldots, N\}$
    - the i-th joint at time t
  - $F(v_{ti})$: node feature, containing coordinate vector, estimation confidence, etc.
  - $E_S = \{v_{ti}v_{tj}\}$: human body joints
    - same t
  - $E_F = \{v_{ti}v_{(t+1)i}\}$: a particular joint i's trajectory over time



[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

Spatial Temporal Graph Convolutional Networks (ST-GCN) [1]

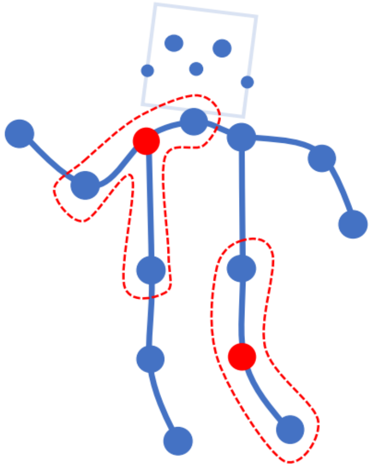
- Let's start from one single frame at t
  - Spatial Graph Convolutional Neural Network

the input feature of v_{tj}

$$f_{out}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{in}(v_{tj}) \cdot \mathbf{w}(v_{ti}, v_{tj})$$

normalizing term: how many number of nodes that are equivalent to v_{tj}, towards v_{ti}

neighbors of v_{ti} $B(v_{ti}) = \{v_{tj} | d(v_{tj}, v_{ti}) \leq D\}$



  - which can be realized by GCN layer [2]

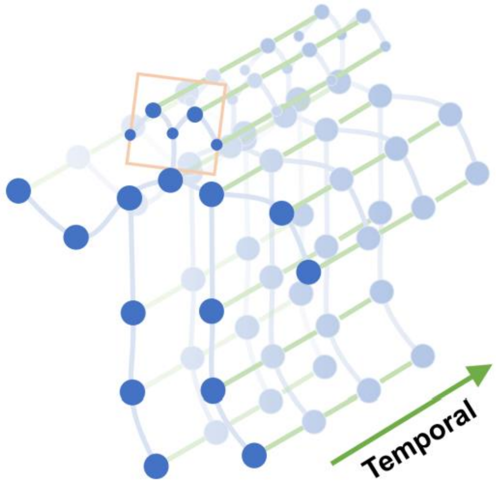
Spatial Temporal Graph Convolutional Networks (ST-GCN) [1]

- Then, let's consider multiple timestamps
  - Recall the Spatial Graph Convolutional Neural Network

$$f_{out}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{in}(v_{tj}) \cdot \mathbf{w}(v_{ti}, v_{tj})$$

$$B(v_{ti}) = \{v_{tj} | d(v_{tj}, v_{ti}) \leq D\}$$

  - For Spatial Temporal Graph Convolution
    - Spatial Temporal Modeling



$$B(v_{ti}) = \{v_{qj} | d(v_{tj}, v_{ti}) \leq K, |q - t| \leq \lfloor \Gamma/2 \rfloor\}$$

a hyperparameter controlling the time range

[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018
[2] Thomas N. Kipf, Max Welling: Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2017

Spatial Temporal Graph Convolutional Networks (ST-GCN) [1]

- A single frame shares the time, i.e.,

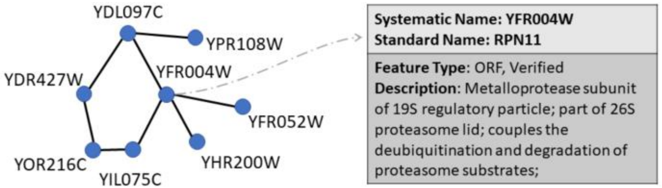
  $E_S = \{v_{ti}v_{tj}\}$: human body joints

$$f_{out}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{in}(v_{tj}) \cdot \mathbf{w}(v_{ti}, v_{tj})$$

intra-snapshot edges

spatial graph convolution for a snapshot

- Is that possible that they have different timestamps?
  - In [2], each dynamic protein-protein interaction network has 36 continuous observations (i.e., **36 edge timestamps**)
  - every 12 observations compose a metabolic cycle (i.e., **3 snapshot timestamps**), and each cycle reflects 25 mins in the real world.



[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018
[2] Dongqi Fu, Jingrui He: DPPIN: A Biological Repository of Dynamic Protein-Protein Interaction Network Data. IEEE Big Data 2022

Given multiple timestamps (e.g., edge timestamps and snapshot timestamps) in a temporal graph

- RQ1: How to integrate the multiple evolution patterns?

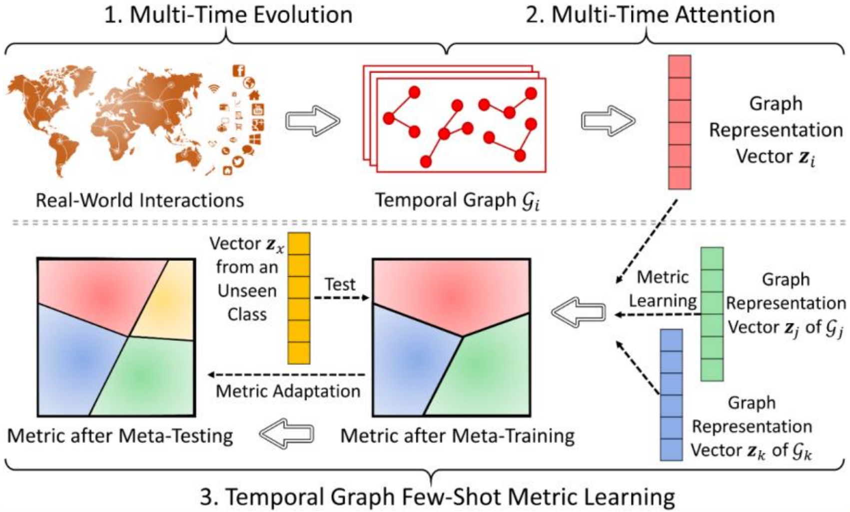
- RQ2: How to encode them for an embedding for temporal graph classification? What evolutions are dominating the graph similarity?

- RQ3: Labeling graph (especially temporal) is costly, how could we leverage fewer labels but effectively?

[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018
[2] Dongqi Fu, Jingrui He: DPPIN: A Biological Repository of Dynamic Protein-Protein Interaction Network Data. IEEE Big Data 2022

Facing above research questions, in Temp-GFSM [1]

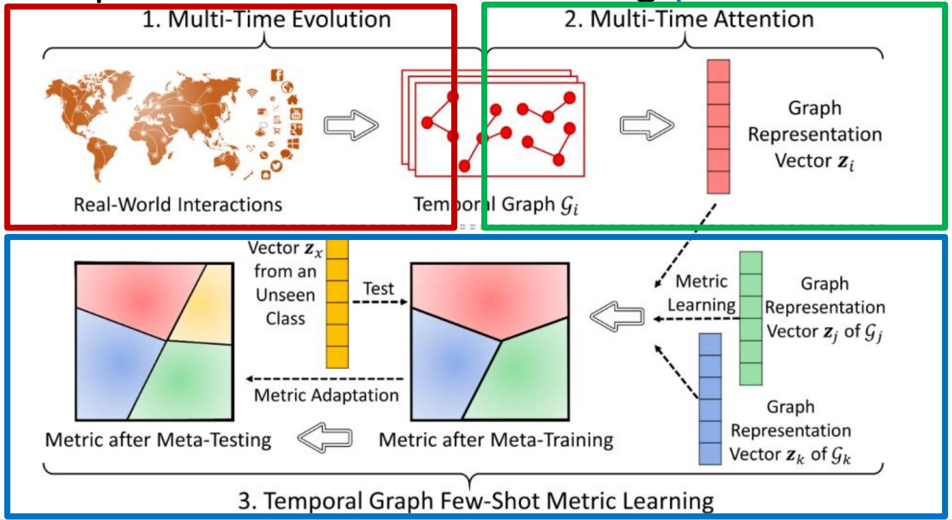
- Multi-Time Evolution
- Multi-Time Attention
- Temporal Graph Few-Shot Metric Learning



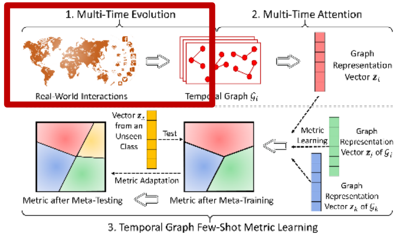
[1] Dongqi Fu, Liri Fang, Ross Maciejewski, Vetle I. Torvik, Jingrui He: Meta-Learned Metrics over Multi-Evolution Temporal Graphs. KDD 2022

In Temp-GFSM [1]

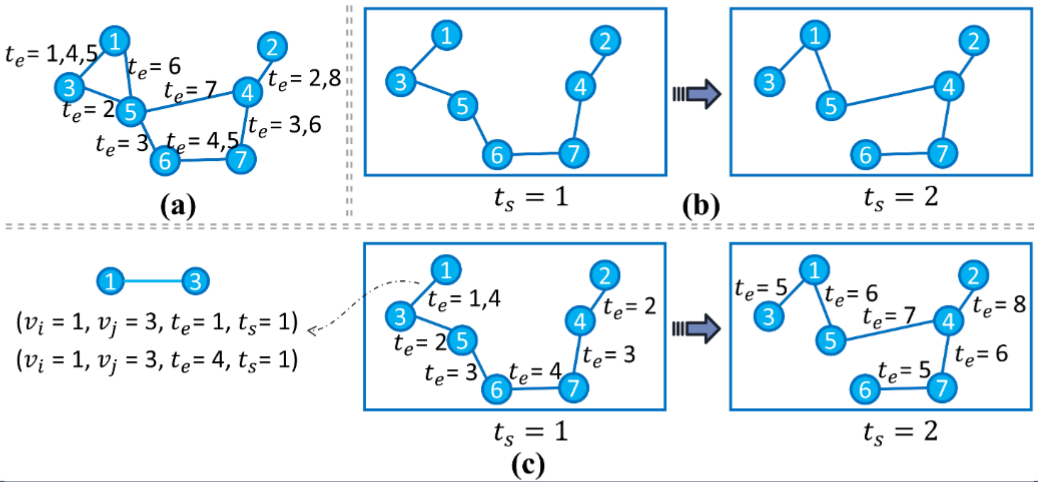
- Multi-Time Evolution (Carrying Multiple Dynamics)
- Multi-Time Attention (Weighting Multiple Dynamics)
- Temporal Graph Few-Shot Metric Learning (New Class Adaption)



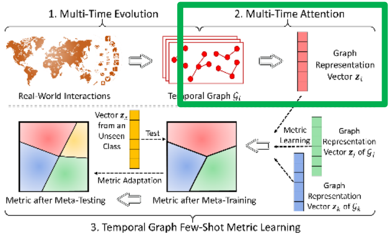
In Multi-Time Evolution of Temp-GFSM [1]



- An edge is marked as quadruplet (v_i, v_j, t_e, t_s), where
  - (v_i, v_j, t_e) means the connection between v_i and v_j exists at time t_e
  - (v_i, v_j, t_e, t_s) means the event (v_i, v_j, t_e) happens in snapshot S^{t_s}

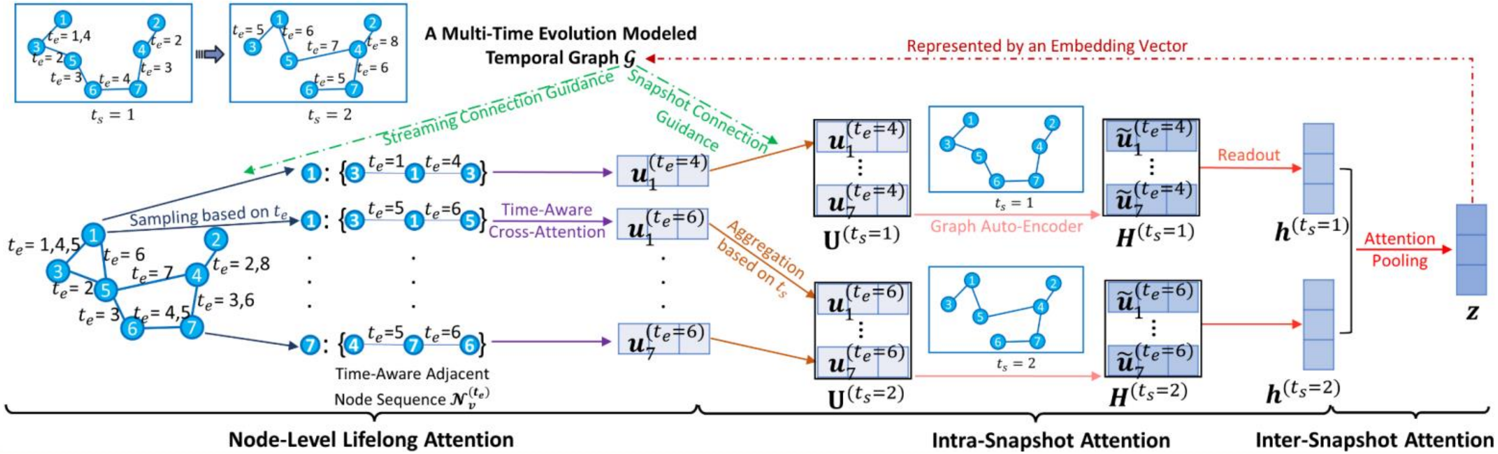


[1] Dongqi Fu, Liri Fang, Ross Maciejewski, Vetle I. Torvik, Jingrui He: Meta-Learned Metrics over Multi-Evolution Temporal Graphs. KDD 2022



In Multi-Time Attention of Temp-GFSM [1]

- Temporal graph G -> representation vector Z
  - Node-Level Lifelong Attention (Select Meaningful Words)
  - Intra-Snapshot Attention (Compose Supportive Sentences)
  - Inter-Snapshot Attention (Finish a Fluent Article with Paragraphs)



[1] Dongqi Fu, Liri Fang, Ross Maciejewski, Vetle I. Torvik, Jingrui He: Meta-Learned Metrics over Multi-Evolution Temporal Graphs. KDD 2022

Spatio-Temporal Graph Convolutional Networks (STGCN) [1]

- **Task:** node-level representation learning
- **Natural dynamic:** evolving node features w.r.t time
- **Goal:** node feature prediction

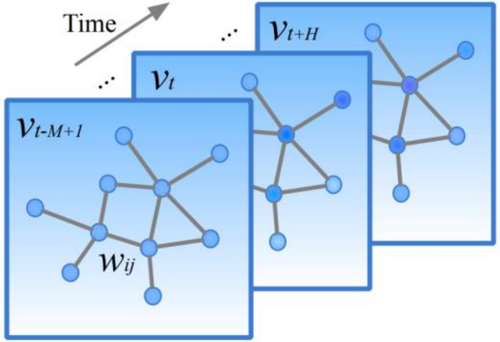
[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

Spatio-Temporal Graph Convolutional Networks (STGCN) [1]

- A Deep Learning Framework for Traffic Forecasting

- Problem Setting
  - Traffic Flow Prediction

given past volumes,
predict future volumes,
with the latent structure



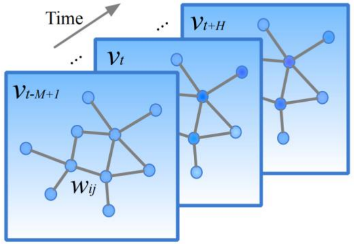
$$\arg\max \quad \log P(v_{t+1}, \ldots, v_{t+H} | v_{t-M+1}, \ldots, v_t)$$

  - $v_t \in \mathbb{R}^n$: an observation of n road segments (n nodes in graph), e.g., volume or density

  - $w \in \mathbb{R}^{n \times n}$: adjacency matrix of road networks, the shared structure over t

[1] Bing Yu, Haoteng Yin, Zhanxing Zhu: Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. IJCAI 2018

Spatio-Temporal Graph Convolutional Networks (STGCN) [1]



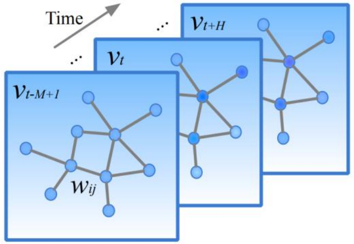
- Extract Spatial Features
  - Similar to previous ST-GCN [2], backbone is GCN
- Extract Temporal Features
  - Other than directly calling GCN on the catenation of temporal features, $v^l = \{v_{t-M+1}, \ldots, v_t\}$, involve a time convolution on the time series as below

$$v^{l+1} = \Gamma_1^l *_{\mathcal{T}} \mathrm{ReLU}(\Theta^l *_{\mathcal{G}} (\Gamma_0^l *_{\mathcal{T}} v^l))$$

$v_t \in \mathbb{R}^n$: an observation of n road segments (n nodes in graph), e.g., volume or density

[1] Bing Yu, Haoteng Yin, Zhanxing Zhu: Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. IJCAI 2018
[2] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

Spatio-Temporal Graph Convolutional Networks (STGCN) [1]



- Extract Spatial Features
  - Similar to previous ST-GCN [2], backbone is GCN
- Extract Temporal Features
  - Other than directly calling GCN on the catenation of temporal features, $v^l = \{v_{t-M+1}, \ldots, v_t\}$, involve a time convolution on the time series as below

$$v^{l+1} = \Gamma_1^l *_{\mathcal{T}} \mathrm{ReLU}(\Theta^l *_{\mathcal{G}} (\Gamma_0^l *_{\mathcal{T}} v^l))$$

l: is the index of unit block (or layer) of STGCN, i.e., $v^l \rightarrow v^{l+1}$

pass a fully-connected output layer to readout v_{t+1}

graph convolution, i.e., GCN

time convolution, (details next page)

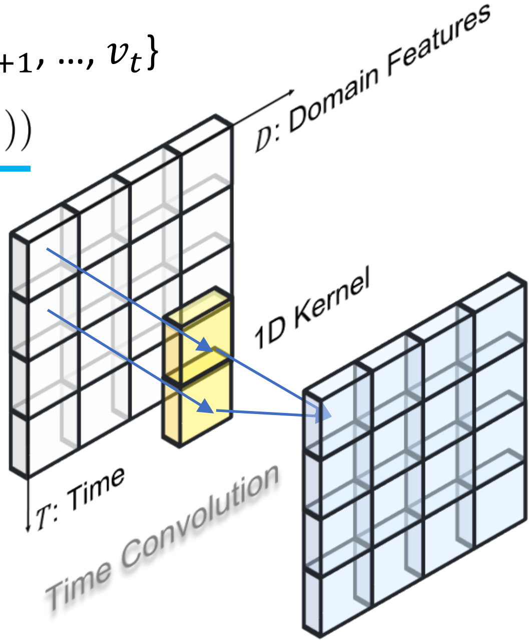
v^l: stacking $v_{t-M+1}, \ldots, v_t$

Spatio-Temporal Graph Convolutional Networks (STGCN) [1]

- Extract Temporal Features
  - Other than directly call GCN on the catenation $v^l = \{v_{t-M+1}, \ldots, v_t\}$

$$\mathrm{ReLU}(\Theta^l *_{\mathcal{G}} (v^l)) \quad \text{VS} \quad \mathrm{ReLU}(\Theta^l *_{\mathcal{G}} (\Gamma_0^l *_{\mathcal{T}} v^l))$$

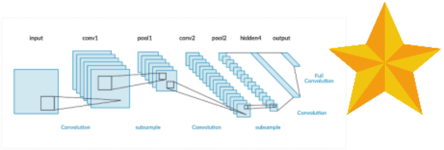
  - A 1-D kernel along the time axis
    - Could aggregate temporal neighbors to **capture temporal behaviors** of features (e.g., traffic flows), especially for long-term time-series



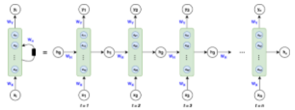
[1] Bing Yu, Haoteng Yin, Zhanxing Zhu: Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. IJCAI 2018
[2] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

Today

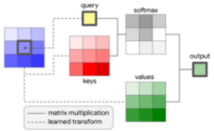
- Covered works for natural dynamics in GNNs
  - Temporal GNNs with Convolutional Operations



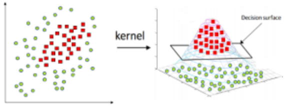
  - Temporal GNNs with Recurrent Units



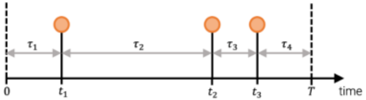
  - Temporal GNNs with Time Attention



  - Temporal GNNs with Time Kernel



  - Temporal GNNs with Temporal Point Process



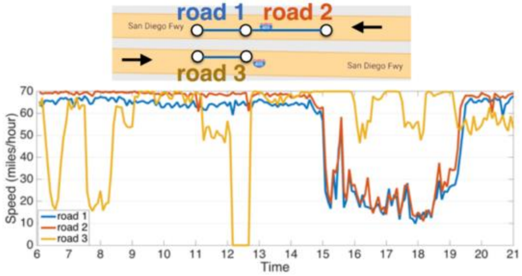
Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- **Task:** node-level representation learning
- **Natural dynamic:** evolving node features w.r.t time
- **Goal:** node feature prediction

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

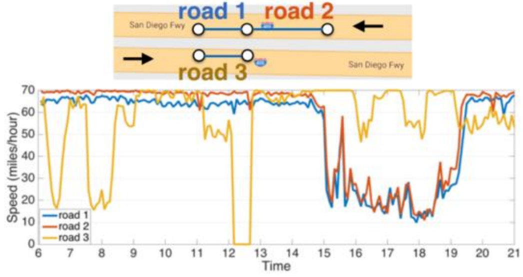
- Problem Definition
  - Let's still focus on Traffic Forecasting



$$[\boldsymbol{X}^{(t-T'+1)}, \cdots, \boldsymbol{X}^{(t)}; \mathcal{G}] \xrightarrow{h(\cdot)} [\boldsymbol{X}^{(t+1)}, \cdots, \boldsymbol{X}^{(t+T)}]$$

  - But the differences from the previous discussed STGCN are:
    - What if the latent graph structure is directed?
    - How can be deal with time information other than time convolution, e.g., how to take time information recurrently?

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]



- For directed graph structure
  - (1) Stationary distribution of the diffusion process

$$\mathcal{P} = \sum_{k=0}^{\infty} \alpha(1-\alpha)^k \left(\boldsymbol{D}_O^{-1}\boldsymbol{W}\right)^k$$

  - (2) Diffusion Convolution

$$\boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\theta}} = \sum_{k=0}^{K-1} \left(\theta_{k,1}\left(\boldsymbol{D}_O^{-1}\boldsymbol{W}\right)^k + \theta_{k,2}\left(\boldsymbol{D}_I^{-1}\boldsymbol{W}^{\intercal}\right)^k\right) \boldsymbol{X}_{:,p} \quad \text{for } p \in \{1, \cdots, P\}$$

  - (3) Diffusion Convolution Layer

$$\boldsymbol{H}_{:,q} = \boldsymbol{a}\left(\sum_{p=1}^{P} \boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\Theta}_{q,p,:,:}}\right) \quad \text{for } q \in \{1, \cdots, Q\}$$

let's step into each one of those

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (1) Stationary distribution of the diffusion process can be represented as a weighted combination of infinite random walks on the graph

$$\mathcal{P} = \sum_{k=0}^{\infty} \alpha(1-\alpha)^k \left(\boldsymbol{D}_O^{-1}\boldsymbol{W}\right)^k$$

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (1) Stationary distribution of the diffusion process can be represented as a weighted combination of infinite random walks on the graph

D_O: out-degree diagonal matrix

W: weighted adjacency matrix

$$\mathcal{P} = \sum_{k=0}^{\infty} \alpha(1-\alpha)^k \left(\boldsymbol{D}_O^{-1}\boldsymbol{W}\right)^k$$

k: random walk steps

$\mathcal{P} \in \mathbb{R}^{N \times N}$: whose i-th row represents the likelihood of diffusion (i.e., personalized PageRank vector) from node i

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (2) Diffusion Convolution **over** a graph signal $\boldsymbol{X} \in \mathbb{R}^{N \times P}$ and a filter f_θ is defined as

$$\boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\theta}} = \sum_{k=0}^{K-1} \left(\theta_{k,1} \left(\boldsymbol{D}_O^{-1} \boldsymbol{W} \right)^k + \theta_{k,2} \left(\boldsymbol{D}_I^{-1} \boldsymbol{W}^\intercal \right)^k \right) \boldsymbol{X}_{:,p} \quad \text{for } p \in \{1, \cdots, P\}$$

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (2) Diffusion Convolution **over** a graph signal $\boldsymbol{X} \in \mathbb{R}^{N \times P}$ and a filter f_θ is defined as

feature matrix, N is the num of node, P is the node feature dimension

learnable parameter, i.e., weight matrices

$$\boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\theta}} = \sum_{k=0}^{K-1} \left(\theta_{k,1} \left(\boldsymbol{D}_O^{-1} \boldsymbol{W} \right)^k + \theta_{k,2} \left(\boldsymbol{D}_I^{-1} \boldsymbol{W}^\intercal \right)^k \right) \boldsymbol{X}_{:,p} \quad \text{for } p \in \{1, \cdots, P\}$$

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (2) Diffusion Convolution **over** a graph signal $\boldsymbol{X} \in \mathbb{R}^{N \times P}$ and a filter f_θ is defined as

feature matrix, N is the num of node, P is the node feature dimension

learnable parameter, i.e., weight matrices

out-degree based diffusion

in-degree based diffusion

$$\boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\theta}} = \sum_{k=0}^{K-1} \left(\theta_{k,1} \left(\boldsymbol{D}_O^{-1} \boldsymbol{W} \right)^k + \theta_{k,2} \left(\boldsymbol{D}_I^{-1} \boldsymbol{W}^{\intercal} \right)^k \right) \boldsymbol{X}_{:,p} \quad \text{for } p \in \{1, \cdots, P\}$$

two sets of parameters from f_θ

feature matrix

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (3) Diffusion Convolution Layer that maps P-dimensional features to Q-dimensional outputs

$$\boldsymbol{H}_{:,q} = \boldsymbol{a}\left(\sum_{p=1}^{P} \boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\Theta}_{q,p,:,:}}\right) \quad \text{for } q \in \{1, \cdots, Q\}$$

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (3) Diffusion Convolution Layer that maps P-dimensional features to Q-dimensional outputs

$\boldsymbol{X} \in \mathbb{R}^{N\times P}$: input

$\boldsymbol{H} \in \mathbb{R}^{N\times Q}$: output

$$\boldsymbol{H}_{:,q} = \boldsymbol{a}\left(\sum_{p=1}^{P} \boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\Theta}_{q,p,:,:}}\right) \quad \text{for } q \in \{1,\cdots,Q\}$$

indexing for in-degree or out-degree diffusion

activation function (e.g., ReLU, Sigmoid)

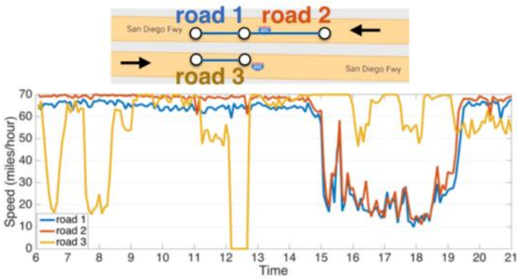
indexing for k

$$\boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\theta}} = \sum_{k=0}^{K-1}\left(\theta_{k,1}\left(\boldsymbol{D}_O^{-1}\boldsymbol{W}\right)^k + \theta_{k,2}\left(\boldsymbol{D}_I^{-1}\boldsymbol{W}^\intercal\right)^k\right)\boldsymbol{X}_{:,p} \quad \text{for } p \in \{1,\cdots,P\}$$

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- After for directed graph structure,
  - (1) Stationary distribution of the diffusion process
  - (2) Diffusion Convolution
  - (3) Diffusion Convolution Layer

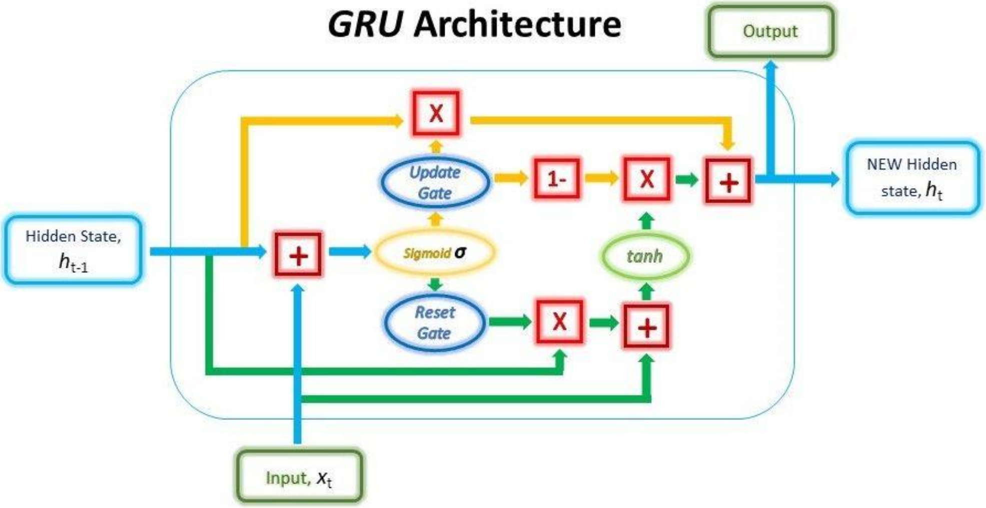


- How to take time information recurrently?
  - Temporal Dynamics Modeling [1]

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

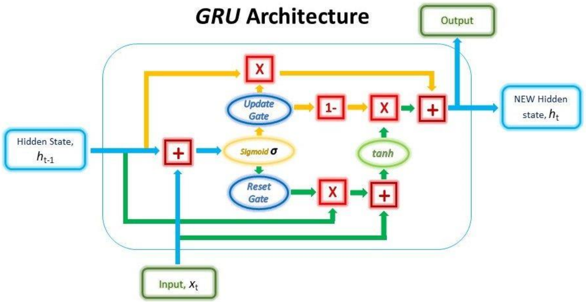
- Temporal Dynamics Modeling
  - Adopt the logic from Gated Recurrent Units (GRU)[2], i.e., make GRU take structured information
  - Input
  - Reset Gate
  - Update Gate
  - Output



[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018
[2] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, Yoshua Bengio: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. CoRR (2014)

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- Temporal Dynamics Modeling
  - Adopt the logic from Gated Recurrent Units (GRU)[2], i.e., make GRU take structured information



  - Input $\boldsymbol{X}^{(t)},\ \boldsymbol{H}^{(t-1)}$
  - Reset Gate $\boldsymbol{r}^{(t)} = \sigma(\boldsymbol{\Theta}_r \star_{\mathcal{G}} [\boldsymbol{X}^{(t)},\ \boldsymbol{H}^{(t-1)}] + \boldsymbol{b}_r)$
  - Update Gate $\boldsymbol{u}^{(t)} = \sigma(\boldsymbol{\Theta}_u \star_{\mathcal{G}} [\boldsymbol{X}^{(t)},\ \boldsymbol{H}^{(t-1)}] + \boldsymbol{b}_u)$

diffusion convolution w. different weight parameters

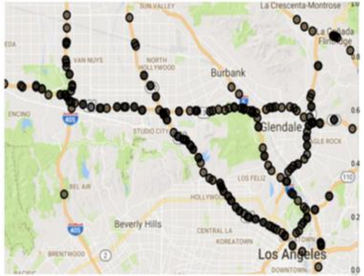
  - Output $\boldsymbol{H}^{(t)} = \boldsymbol{u}^{(t)} \odot \boldsymbol{H}^{(t-1)} + (1 - \boldsymbol{u}^{(t)}) \odot \boldsymbol{C}^{(t)}$

$$\boldsymbol{C}^{(t)} = \tanh(\boldsymbol{\Theta}_C \star_{\mathcal{G}} [\boldsymbol{X}^{(t)},\ (\boldsymbol{r}^{(t)} \odot \boldsymbol{H}^{(t-1)})] + \boldsymbol{b}_c)$$

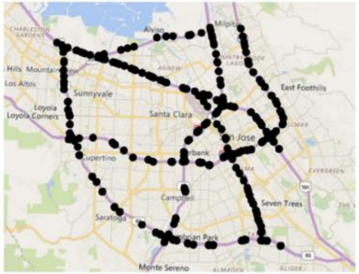
[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018
[2] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, Yoshua Bengio: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. CoRR (2014)

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- Experiments
  - Datasets
    - METR-LA: Highway of Los Angeles
      - 207 sensors (traffic speed)
      - May 1st 2012 to Jun 30th 2012
    - PEMS-BAY: Highway in Bay Area of California
      - 325 sensors (traffic speed)
      - Jan 1st 2017 to May 31th 2017



(a) METR-LA



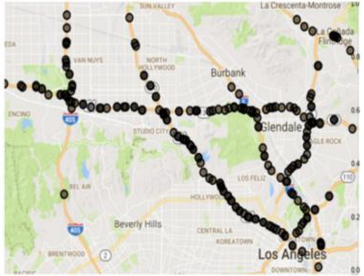
(b) PEMS-BAY

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

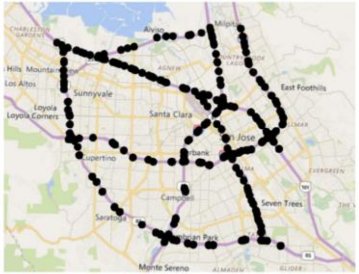
Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- Experiments
  - Performance (traffic speed forecasting)

	T	Metric	HA	ARIMA$_{Kal}$	VAR	SVR	FNN	FC-LSTM	*DCRNN*
METR-LA	15 min	MAE	4.16	3.99	4.42	3.99	3.99	3.44	**2.77**
		RMSE	7.80	8.21	7.89	8.45	7.94	6.30	**5.38**
		MAPE	13.0%	9.6%	10.2%	9.3%	9.9%	9.6%	**7.3%**
	30 min	MAE	4.16	5.15	5.41	5.05	4.23	3.77	**3.15**
		RMSE	7.80	10.45	9.13	10.87	8.17	7.23	**6.45**
		MAPE	13.0%	12.7%	12.7%	12.1%	12.9%	10.9%	**8.8%**
	1 hour	MAE	4.16	6.90	6.52	6.72	4.49	4.37	**3.60**
		RMSE	7.80	13.23	10.11	13.76	8.69	8.69	**7.59**
		MAPE	13.0%	17.4%	15.8%	16.7%	14.0%	13.2%	**10.5%**
PEMS-BAY	15 min	MAE	2.88	1.62	1.74	1.85	2.20	2.05	**1.38**
		RMSE	5.59	3.30	3.16	3.59	4.42	4.19	**2.95**
		MAPE	6.8%	3.5%	3.6%	3.8%	5.19%	4.8%	**2.9%**
	30 min	MAE	2.88	2.33	2.32	2.48	2.30	2.20	**1.74**
		RMSE	5.59	4.76	4.25	5.18	4.63	4.55	**3.97**
		MAPE	6.8%	5.4%	5.0%	5.5%	5.43%	5.2%	**3.9%**
	1 hour	MAE	2.88	3.38	2.93	3.28	2.46	2.37	**2.07**
		RMSE	5.59	6.50	5.44	7.08	4.98	4.96	**4.74**
		MAPE	6.8%	8.3%	6.5%	8.0%	5.89%	5.7%	**4.9%**



(a) METR-LA



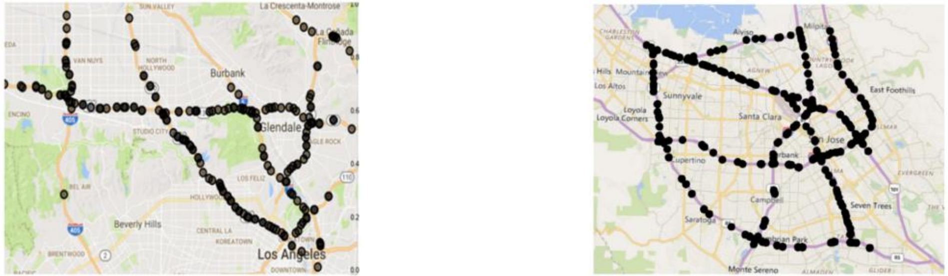
(b) PEMS-BAY

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

The latent graph structure may be not easy to observe

- In DCRNN[1], the adjacency is hand-crafted

$$W_{ij} = \exp\left(-\frac{\text{dist}(v_i, v_j)^2}{\sigma^2}\right)$$



(a) METR-LA

(b) PEMS-BAY

- Could we find another way to extract that latent structure?

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Could we find another way to extract that latent structure?

- Discrete Graph Structure Learning for Forecasting Multiple Time Series (GTS) [2]

  - Focusing on the same problem (i.e., traffic forecasting) and the same diffusion convolution structure as DCRNN [1]

$$[\boldsymbol{X}^{(t-T'+1)}, \cdots, \boldsymbol{X}^{(t)}; \mathcal{G}] \xrightarrow{h(\cdot)} [\boldsymbol{X}^{(t+1)}, \cdots, \boldsymbol{X}^{(t+T)}]$$

  - But set the adjacency matrix as a variable to learn

$$A_{ij} = \text{sigmoid}((\log(\theta_{ij}/(1-\theta_{ij})) + (g_{ij}^1 - g_{ij}^2))/s)$$

temperature

samples from a given Gumbel distribution

$$\theta_{ij} = \text{FC}(\text{FC}(z^i \| z^j))$$

$$z^i = \text{FC}(\text{vec}(\text{Conv}(X^i)))$$

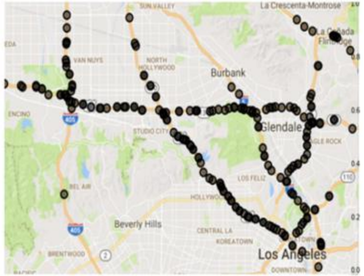
X^i: the i-th node over all features and timestamps

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018
[2] Chao Shang, Jie Chen, Jinbo Bi: Discrete Graph Structure Learning for Forecasting Multiple Time Series. ICLR 2021

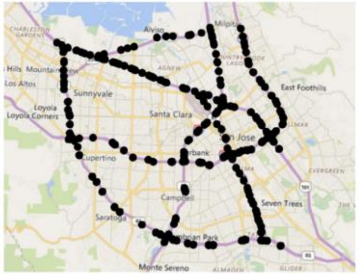
In the same datasets with [1], i.e., METR-LA and PEMS-BAY

- Experiments
  - Performance (traffic speed forecasting)

	Metric	FNN	LSTM	DCRNN	LDS	NRI	GTSv	GTS
15 min	MAE ($\times 10^{-3}$)	1.23	1.02	0.71	0.49	0.66	0.26	**0.24**
	RMSE ($\times 10^{-2}$)	1.28	1.63	1.42	1.26	0.27	0.20	**0.19**
	MAPE	0.20%	0.21%	0.09%	0.07%	0.14%	0.05%	**0.04%**
30 min	MAE ($\times 10^{-3}$)	1.42	1.11	1.08	0.81	0.71	0.31	**0.30**
	RMSE ($\times 10^{-2}$)	1.81	2.06	1.91	1.79	0.30	0.23	**0.22**
	MAPE	0.23%	0.20%	0.15%	0.12%	0.15%	**0.05%**	**0.05%**
60 min	MAE ($\times 10^{-3}$)	1.88	1.79	1.78	1.45	0.83	**0.39**	0.41
	RMSE ($\times 10^{-2}$)	2.58	2.75	2.65	2.54	0.46	0.32	**0.30**
	MAPE	0.29%	0.27%	0.24%	0.22%	0.17%	**0.07%**	**0.07%**



(a) METR-LA

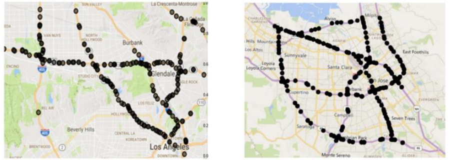


(b) PEMS-BAY

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018
[2] Chao Shang, Jie Chen, Jinbo Bi: Discrete Graph Structure Learning for Forecasting Multiple Time Series. ICLR 2021

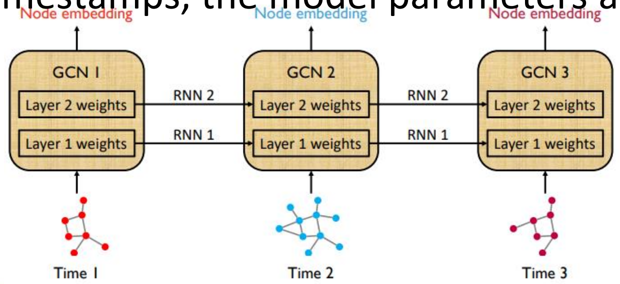
In DCRNN [1] or GTS [2],

- The adjacency is fixed with evolving node features
- What if the adjacency is also evolving?



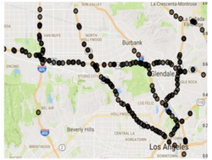
(a) METR-LA (b) PEMS-BAY

- For evolving structures and features
  - EvolveGCN [3] is proposed to adapt model parameters, i.e.,
    - Each time has its own GCN model with its $A^{(t)}$ and $H^{(t)}$
    - Cross timestamps, the model parameters are dependent, e.g., $W_t^{(l)} = LSTM(W_{t-1}^{(l)})$

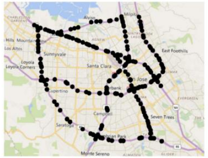


[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018
[2] Chao Shang, Jie Chen, Jinbo Bi: Discrete Graph Structure Learning for Forecasting Multiple Time Series. ICLR 2021
[3] Pareja et al.: EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. AAAI 2020

In DCRNN [1] or GTS [2],

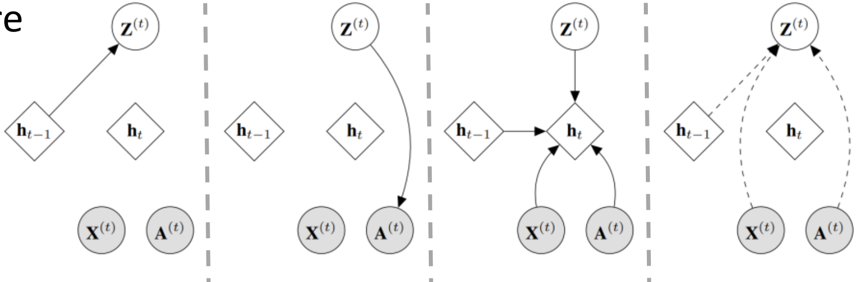


(a) METR-LA


(b) PEMS-BAY

- The adjacency is fixed with evolving node features
- ~~What if the adjacency is also evolving?~~
- Can we also predict the future adjacency?

- For evolving structures and features
  - VGRNN [3] is proposed to learn the variational posterior distribution of evolving adjacency structures together, in the RNN structure

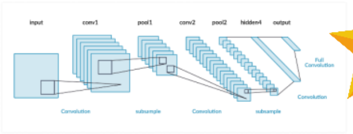


(a) Prior (b) Generation (c) Recurrence (d) Inference

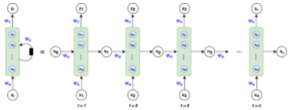
[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018
[2] Chao Shang, Jie Chen, Jinbo Bi: Discrete Graph Structure Learning for Forecasting Multiple Time Series. ICLR 2021
[3] Hajiramezanali et al.: Variational Graph Recurrent Neural Networks. NeurIPS 2019

In this tutorial

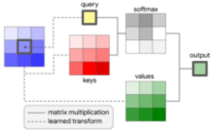
- Covered works for natural dynamics in GNNs
  - Temporal GNNs with Convolutional Operations



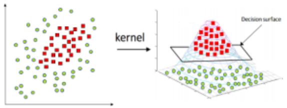
  - Temporal GNNs with Recurrent Units



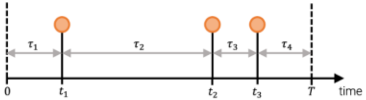
  - Temporal GNNs with Time Attention



  - Temporal GNNs with Time Kernel



  - Temporal GNNs with Temporal Point Process



DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- **Task:** node-level representation learning
- **Natural dynamic:** evolving graph structures and node features w.r.t time
- **Goal:** link prediction

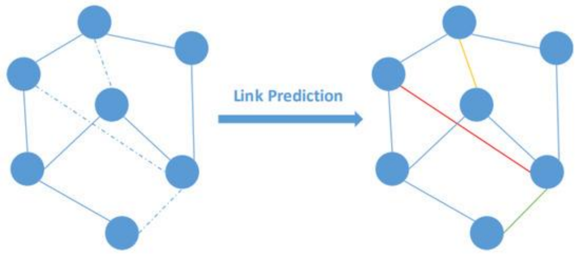
[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Problem Definition (Link Prediction)

  - Given a series of graph snapshots $\{\mathcal{G}^1, \ldots, \mathcal{G}^T\}$, and $\mathcal{G}^t = (\boldsymbol{A}^t, \boldsymbol{X}^t)$, DySAT [1] aims to learn the node representation e_v^t for each node v at timestamps $t = \{1, 2, \ldots, T\}$

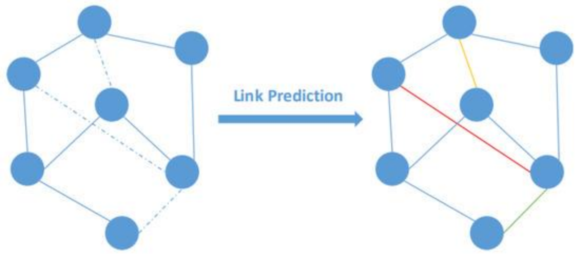
  - Then, the latest time e_v^T and e_u^T are used to decide if there is an edge links node v and node u at time T+1



DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Structural Self-Attention
  - Apply attention in one single timestamp
  - Topological neighbors

- Temporal Self-Attention
  - Apply attention across timestamps
  - Temporal neighbors



[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Structural Self-Attention
  - At a single timestamp t, *the superscript of t is omitted in the following equation*

$\boldsymbol{z}_v$: hidden representation of node v after structural attention

a_{uv}: attention weight

$$z_v = \sigma\Big(\sum_{u \in \mathcal{N}_v} \alpha_{uv} \boldsymbol{W}^s \boldsymbol{x}_u\Big), \quad \alpha_{uv} = \frac{\exp(e_{uv})}{\sum_{w \in \mathcal{N}_v} \exp(e_{wv})}$$

A_{uv}: adjacency

neighbors in $\boldsymbol{A}^t$

$$e_{uv} = \sigma\Big(A_{uv} \cdot \boldsymbol{a}^T [\boldsymbol{W}^s \boldsymbol{x}_u || \boldsymbol{W}^s \boldsymbol{x}_v]\Big) \quad \forall (u, v) \in \mathcal{E}$$

$\boldsymbol{a}$: learnable weight matrix (e.g., MLP)

$\boldsymbol{W}^s$: projection weight matrix for structural attention

[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Temporal Self-Attention
  - Who are temporal neighbors for a certain node?
  - Temporal neighbors for node v consist of its **historical behaviors**

[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Temporal Self-Attention
  - Temporal neighbors for node v consist of its historical behaviors

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

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

Node v's time embedding, $\mathbb{R}^{T\times F'}$

$\mathbb{R}^{T\times T}$

$\mathbb{R}^{T\times D'}$

$\mathbb{R}^{D'\times F'}$

attention weights, i and j are two timestamps

X_v: concatenation of $x_v^1, x_v^2, \ldots, x_v^T$

F': normalization factor

A matrix enforcing auto-regressive manner, details next page

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Temporal Self-Attention
    - Temporal neighbors for node v are its historical behaviors

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

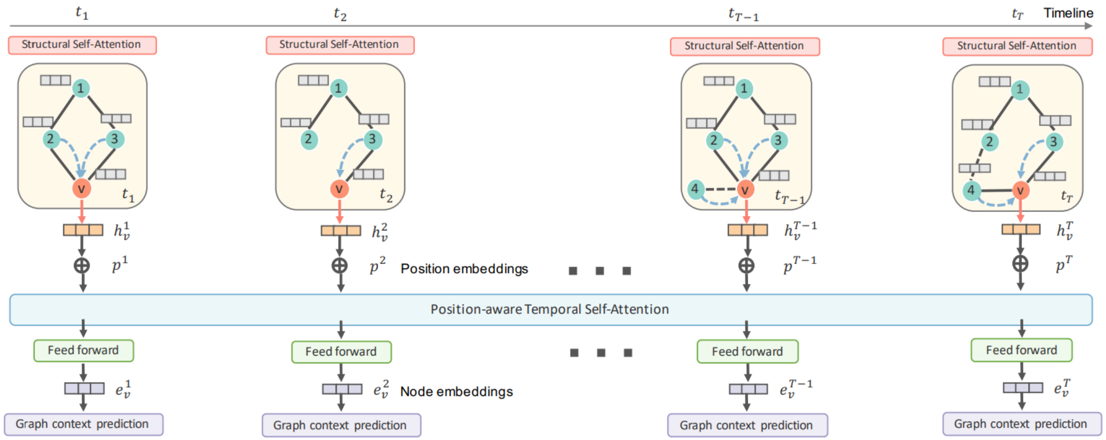
$$e_v^{ij} = \left(\frac{((\mathbf{X}_v \mathbf{W}_q)(\mathbf{X}_v \mathbf{W}_k)^T)_{ij}}{\sqrt{F'}} + M_{ij} \right)$$

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

when $M_{ij} = -\infty$, $\beta_v^{ij} = 0$, which switches off the attention from timestamp i to j

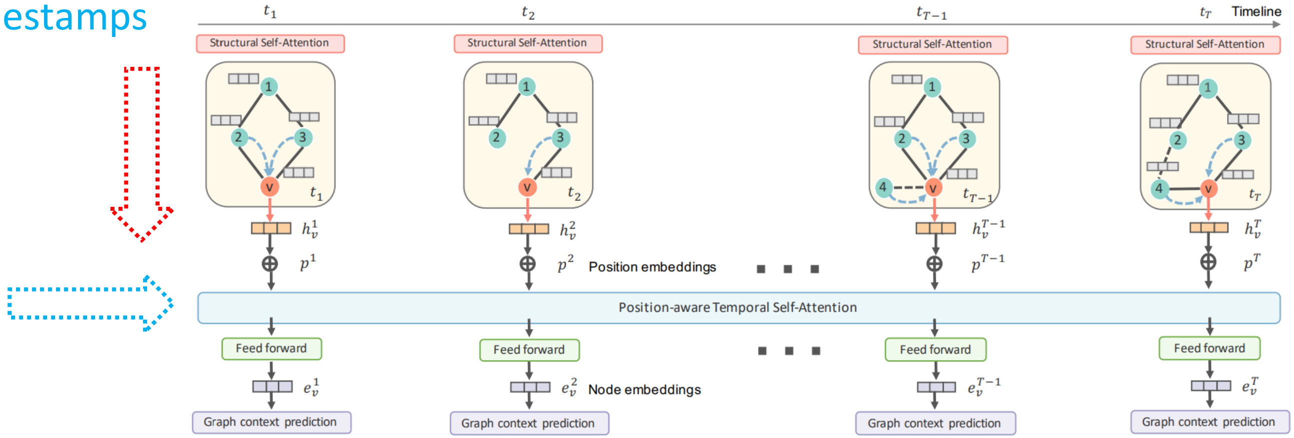
DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Structural + Temporal Self-Attention
  - Obtain structural encoding independently at each timestamp t
  - Then, temporal self-attention take the structural encoding as input to attend over timestamps



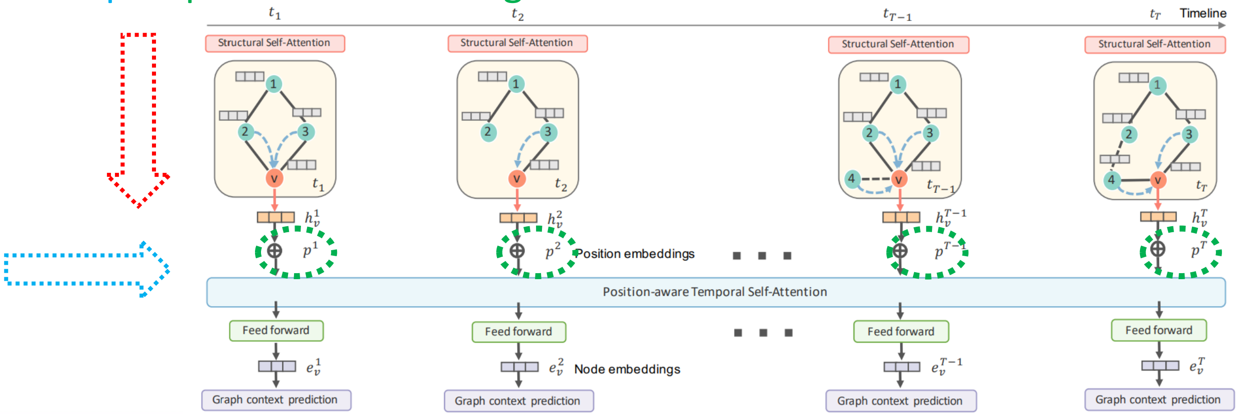
DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Structural + Temporal Self-Attention
  - Obtain structural encoding independently at each timestamp t
  - Then, temporal self-attention take the structural encoding as input to attend over timestamps



DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

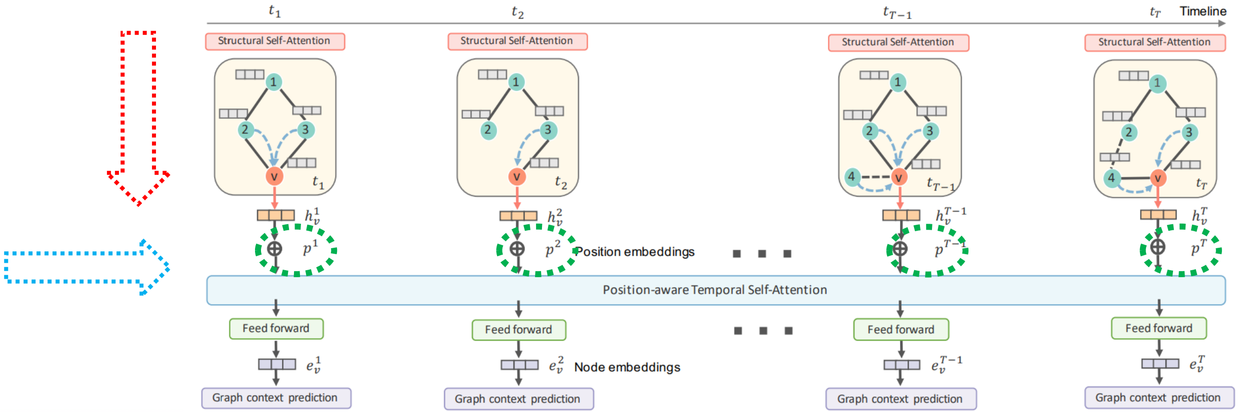
- Structural + Temporal Self-Attention
  - Obtain structural encoding independently at each timestamp t
  - Then, temporal self-attention take the structural encoding as input to attend over timestamps + positional encoding



DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Structural + Temporal Self-Attention
  - Add the absolute temporal position of each snapshot

$$\boldsymbol{h}_v^1 + \boldsymbol{p}^1, \boldsymbol{h}_v^2 + \boldsymbol{p}^2, \ldots, \boldsymbol{h}_v^T + \boldsymbol{p}^T$$



DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Positional Encoding

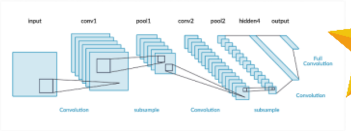
I	0	P_{00}=sin(0) = 0	P_{01}=cos(0) = 1	P_{02}=sin(0) = 0	P_{03}=cos(0) = 1
am	1	P_{10}=sin(1/1) = 0.84	P_{11}=cos(1/1) = 0.54	P_{12}=sin(1/10) = 0.10	P_{13}=cos(1/10) = 1.0
a	2	P_{20}=sin(2/1) = 0.91	P_{21}=cos(2/1) = -0.42	P_{22}=sin(2/10) = 0.20	P_{23}=cos(2/10) = 0.98
Robot	3	P_{30}=sin(3/1) = 0.14	P_{31}=cos(3/1) = -0.99	P_{32}=sin(3/10) = 0.30	P_{33}=cos(3/10) = 0.96

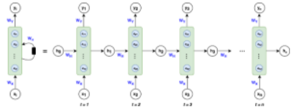
image source: https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

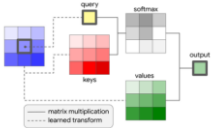
[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

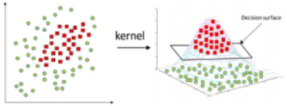
In this tutorial

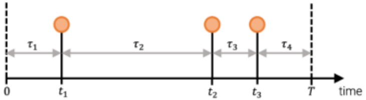
- Covered works for natural dynamics in GNNs
  - Temporal GNNs with Convolutional Operations
  - Temporal GNNs with Recurrent Units
  - Temporal GNNs with Time Attention
  - Temporal GNNs with Time Kernel
  - Temporal GNNs with Temporal Point Process











Back to the positional encoding

I	0	P_{00}=sin(0) = 0	P_{01}=cos(0) = 1	P_{02}=sin(0) = 0	P_{03}=cos(0) = 1
am	1	P_{10}=sin(1/1) = 0.84	P_{11}=cos(1/1) = 0.54	P_{12}=sin(1/10) = 0.10	P_{13}=cos(1/10) = 1.0
a	2	P_{20}=sin(2/1) = 0.91	P_{21}=cos(2/1) = -0.42	P_{22}=sin(2/10) = 0.20	P_{23}=cos(2/10) = 0.98
Robot	3	P_{30}=sin(3/1) = 0.14	P_{31}=cos(3/1) = -0.99	P_{32}=sin(3/10) = 0.30	P_{33}=cos(3/10) = 0.96

- Do we have other options?
  - A concurrent method [1] with DySAT [2] proposes the time kernel function to record the time features

[1] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, Kannan Achan: Inductive representation learning on temporal graphs. ICLR 2020
[2] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

Inductive Representation Learning on Temporal Graphs (TGAT) [1]

- **Task:** node-level representation learning
- **Natural dynamic:** evolving graph structures and node features w.r.t time
- **Goal:** link prediction, node classification

[1] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, Kannan Achan: Inductive representation learning on temporal graphs. ICLR 2020

Inductive Representation Learning on Temporal Graphs (TGAT) [1]

- In [1], **kernel function** is proposed to map time t to a finite dimensional representation vector

$$t \mapsto \Phi_d(t) := \sqrt{\frac{1}{d}}\left[\cos(\omega_1 t), \sin(\omega_1 t), \ldots, \cos(\omega_d t), \sin(\omega_d t)\right]$$

$\omega_1, \omega_2, \ldots, \omega_d$ are learnable parameters

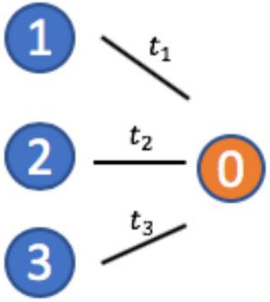
[1] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, Kannan Achan: Inductive representation learning on temporal graphs. ICLR 2020

Inductive Representation Learning on Temporal Graphs (TGAT) [1]

- Suppose there is a target node v_0 at time t, which needs to attend over its spatial-temporal neighbors

$$\mathcal{N}(v_0; t) = \{v_1, \ldots, v_N\}$$

- For each node v_i, v_0 connects with it previously at a time t_i, i.e., $t_i < t$



[1] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, Kannan Achan: Inductive representation learning on temporal graphs. ICLR 2020

Inductive Representation Learning on Temporal Graphs (TGAT) [1]

- With $\mathcal{N}(v_0; t) = \{v_1, \dots, v_N\},\ t_i < t$

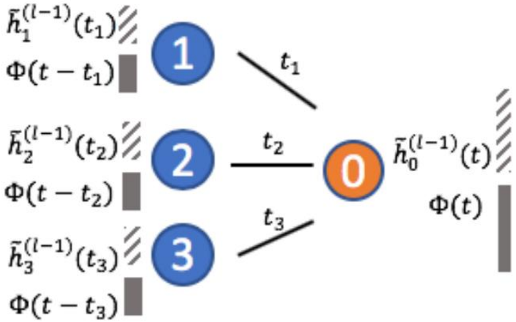
- First, append the position encoding by time kernel functions, to form $\boldsymbol{Z}(t)$

$$\mathbf{Z}(t) = \left[\tilde{\mathbf{h}}_0^{(l-1)}(t) || \Phi_{d_T}(0), \tilde{\mathbf{h}}_1^{(l-1)}(t_1) || \Phi_{d_T}(t - t_1), \dots, \tilde{\mathbf{h}}_N^{(l-1)}(t_N) || \Phi_{d_T}(t - t_N)\right]^{\mathsf{T}}$$

$\boldsymbol{Z}(t)$ is the intermediate step of getting the node embedding $\tilde{h}_0^{(l)}$ of v_0 at the l-th layer of TGAT

node embedding of v_0 at the (l-1)-th layer of TGAT, the first layer is the initial input feature

time kernel function



Inductive Representation Learning on Temporal Graphs (TGAT) [1]

- With $\mathcal{N}(v_0;t) = \{v_1, \ldots, v_N\},\ t_i < t$

- Append the position encoding by time kernel functions

$$\mathbf{Z}(t) = \left[\tilde{\mathbf{h}}_0^{(l-1)}(t)||\Phi_{d_T}(0), \tilde{\mathbf{h}}_1^{(l-1)}(t_1)||\Phi_{d_T}(t-t_1), \ldots, \tilde{\mathbf{h}}_N^{(l-1)}(t_N)||\Phi_{d_T}(t-t_N)\right]^{\intercal}$$

- Self-Attention

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

- Readout

$$\tilde{\mathbf{h}}_0^{(l)}(t) = \mathrm{FFN}\big(\mathbf{h}(t)||\mathbf{x}_0\big) \equiv \mathrm{ReLU}\big([\mathbf{h}(t)||\mathbf{x}_0]\mathbf{W}_0^{(l)} + \mathbf{b}_0^{(l)}\big)\mathbf{W}_1^{(l)} + \mathbf{b}_1^{(l)}$$

[1] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, Kannan Achan: Inductive representation learning on temporal graphs. ICLR 2020

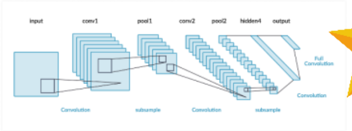
Other Time Kernel Functions [1]

Feature maps specified by $[\phi_{2i}(t), \phi_{2i+1}(t)]$	Origin	Parameters	Interpretations of ω
$\left[\cos\left(\omega_i(\mu)t\right), \sin\left(\omega_i(\mu)t\right)\right]$	Bochner's	μ: location-scale parameters specified for the *reparametrization trick*.	$\omega_i(\mu)$: converts the i^{th} sample (drawn from auxiliary distribution) to target distribution under location-scale parameter μ.
$\left[\cos\left(g_\theta(\omega_i)t\right), \sin\left(g_\theta(\omega_i)t\right)\right]$	Bochner's	θ: parameters for the inverse CDF $F^{-1} = g_\theta$.	ω_i: the i^{th} sample drawn from the auxiliary distribution.
$\left[\cos(\tilde{\omega}_i t), \sin(\tilde{\omega}_i t)\right]$	Bochner's	$\{\tilde{\omega}\}_{i=1}^{d}$: transformed samples under non-parametric inverse CDF transformation.	$\tilde{\omega}_i$: the i^{th} sample of the underlying distribution $p(\omega)$ in Bochner's Theorem.
$\left[\sqrt{c_{2i,k}}\cos(\omega_j t), \sqrt{c_{2i+1,k}}\sin(\omega_j t)\right]$	Mercer's	$\{c_{i,k}\}_{i=1}^{2d}$: the Fourier coefficients of corresponding $\mathcal{K}_{\omega_j}$, for $j = 1, \ldots, k$.	ω_j: the frequency for kernel function $\mathcal{K}_{\omega_j}$ (can be parameters).

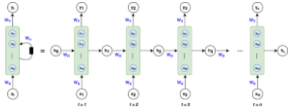
[1] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, Kannan Achan: Self-attention with Functional Time Representation Learning. NeurIPS 2019

Today

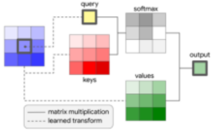
- Covered works for natural dynamics in GNNs
  - Temporal GNNs with Convolutional Operations



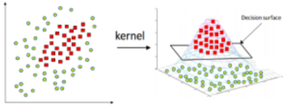
  - Temporal GNNs with Recurrent Units



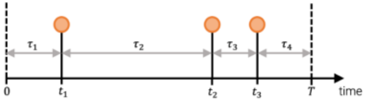
  - Temporal GNNs with Time Attention



  - Temporal GNNs with Time Kernel



  - Temporal GNNs with Temporal Point Process



DyRep: Learning Representations over Dynamic Graphs [1]

- **Task:** node-level representation learning
- **Natural dynamic:** evolving graph structures and node features w.r.t time
- **Goal:** link prediction

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019

DyRep: Learning Representations over Dynamic Graphs [1]

- Temporal Point Process (TPP) [2]
  - A user is tweeting, they tweeted at time t_1= 8:00 am, t_2=10:00 am, t_3= 11:00 am, what is t_4 = ?

  - TPP is a model that could fit the process of t_1, t_2, and t_3 to predict t_4

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019
[2] Upadhyay et al., Temporal Point Processes: https://courses.mpi-sws.org/hcml-ws18/lectures/TPP.pdf

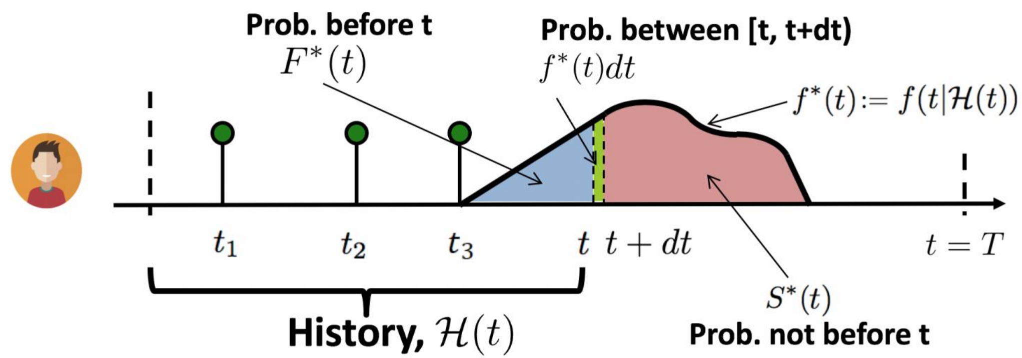
DyRep: Learning Representations over Dynamic Graphs [1]

- Temporal Point Process (TPP) [2]
  - Given the history of events $\mathcal{H}(t)$ = $\{t_1, \ldots, t_{i-1}\}$, we need to model,
    - A conditional probability density function $f^* = f(t|\mathcal{H}(t))$, which is the conditional probability that the next event t will occur during the interval $[t, t+dt)$
    - A cumulative distribution function $F^*(t) = F(t|\mathcal{H}(t))$ = $\int_{t_{i-1}}^{t} f^*(\tau)d\tau$, which is the conditional probability that the next event will occur before t
    - A complementary of $F^*(t)$, $S^*(t) = S(t|\mathcal{H}(t))$ = 1 - $F^*(t)$, the conditional probability that the next event will not occur before time t

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019
[2] Upadhyay et al., Temporal Point Processes: https://courses.mpi-sws.org/hcml-ws18/lectures/TPP.pdf

DyRep: Learning Representations over Dynamic Graphs [1]

- Temporal Point Process (TPP) [2]
  - Given the history $\mathcal{H}(t) = \{t_1, \ldots, t_{i-1}\}$, we need to model,
    - $f^*(t) = f(t|\mathcal{H}(t))$: next event t will occur during the interval $[t, t+dt)$
    - $F^*(t) = F(t|\mathcal{H}(t)) = \int_{t_{i-1}}^{t} f^*(\tau)d\tau$: next event will occur before t
    - $S^*(t) = S(t|\mathcal{H}(t)) = 1 - F^*(t)$: next event will not occur before time t

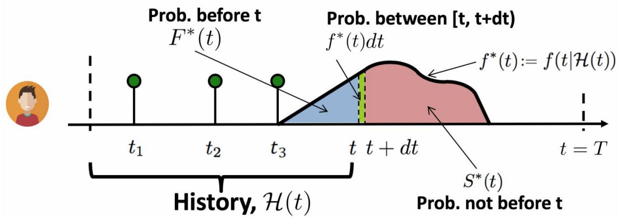


[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019
[2] Upadhyay et al., Temporal Point Processes: https://courses.mpi-sws.org/hcml-ws18/lectures/TPP.pdf

DyRep: Learning Representations over Dynamic Graphs [1]

- Temporal Point Process (TPP) [2]
  - The conditional intensity function $\lambda^*(t) = \lambda(t|\mathcal{H}(t))$, i.e., the conditional probability that the next event will happened during $[t, t{+}dt)$, is defined as follows

$$\lambda^*(t)dt = \frac{f^*(t)dt}{S^*(t)}$$



  - $\lambda^*(t)$ can be also understood as the instantaneous rate of events per time of unit, e.g., $\lambda^*(t)$ = 10 tweets/minute
  - Using the form of $\lambda^*(t)$ also contributes to TPP model parameterization and model reusability [2]

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019
[2] Upadhyay et al., Temporal Point Processes: https://courses.mpi-sws.org/hcml-ws18/lectures/TPP.pdf

DyRep: Learning Representations over Dynamic Graphs [1]

- Temporal Point Process (TPP) [2]
  - Different forms of functions model the intensity function $\lambda^*(t)$, e.g.,
    - Homogeneous Poisson process
      - $\lambda^*(t) = \mu \geq 0$ 10 tweets/minute
    - Inhomogeneous Poisson process
      - $\lambda^*(t) = g_\theta(t) \geq 0$ 2 tweets/@8:35am, 25 tweets/@2:58pm, …
    - Hawkes process
      - $\lambda^*(t) = \mu + \alpha \sum_{t_i \in \mathcal{H}(t)} \kappa_\omega(t - t_i),\ \kappa_\omega(t) = \exp(-\omega t)$
  - The parameters are obtained by fitting the model with the observation and maximizing the log-likelihood

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019
[2] Upadhyay et al., Temporal Point Processes: https://courses.mpi-sws.org/hcml-ws18/lectures/TPP.pdf

DyRep: Learning Representations over Dynamic Graphs [1]

- Model Temporal Point Process (TPP) for Graphs
  - Each **edge connection** is considering as an **event** (u, v, t)
  - We want to predict whether a node u and a node v will connect at time t, given node u's history and node v's history before t

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019

DyRep: Learning Representations over Dynamic Graphs [1]

- Model Temporal Point Process (TPP) for Graphs
  - Intensity functions for graphs, i.e., an edge connection between nodes u and v

$$\lambda^{u,v}(t) = f(g^{u,v}(\bar{t}))$$

$\bar{t}$ means the timestamp just before the current event

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019

DyRep: Learning Representations over Dynamic Graphs [1]

- Model Temporal Point Process (TPP) for Graphs
  - Intensity functions for graphs, i.e., an edge connection between nodes u and v

$$\lambda^{u,v}(t) = f(g^{u,v}(\bar{t}))$$

the outer function $f()$ is a softplus function with trainable a parameter to make sure positive output

$$f(x) = \psi \log(1 + \exp(x/\psi))$$

the inner function $g()$ computes the compatibility of the catenation

$$g^{u,v}(\bar{t}) = \boldsymbol{\omega}\,[z^u(\bar{t}); z^v(\bar{t})]$$

  - Now, the question is how to get the node embeddings, e.g., $z^v(\bar{t})$?

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019

DyRep: Learning Representations over Dynamic Graphs [1]

- Model Temporal Point Process (TPP) for Graphs
  - How to get node embeddings, e.g., $z^v(\bar{t})$?
    - Self-Propagation: w.r.t its historical behavior
    - Exogeneous Drive: for the smooth update of the current
    - Localized Embedding Propagation: message passing within second-order proximity
  - Suppose node u and node v participating in any type of event at time t
    - E.g., for the p-th event of node v at time t

$$\mathbf{z}^v(t_p) = \sigma\Big(\underbrace{\mathbf{W}^{struct}\mathbf{h}^u_{struct}(\bar{t_p})}_{\text{Localized Embedding Propagation}} + \underbrace{\mathbf{W}^{rec}\mathbf{z}^v(\bar{t^v_p})}_{\text{Self-Propagation}} + \underbrace{\mathbf{W}^t(t_p - \bar{t^v_p})}_{\text{Exogenous Drive}}\Big)$$

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019

DyRep: Learning Representations over Dynamic Graphs [1]

- Suppose node u and node v participating in any type of event at time t
  - For the p-th event of node v at time t

$$\mathbf{z}^v(t_p) = \sigma(\underbrace{\mathbf{W}^{struct}\mathbf{h}^u_{struct}(\bar{t_p})}_{\text{Localized Embedding Propagation}} + \underbrace{\mathbf{W}^{rec}\mathbf{z}^v(\bar{t^v_p})}_{\text{Self-Propagation}} + \underbrace{\mathbf{W}^t(t_p - \bar{t^v_p})}_{\text{Exogenous Drive}})$$

h^u_{struc} is the aggregation on node u's neighbors

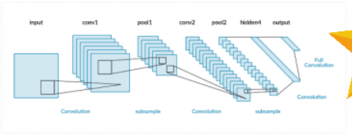
$$h^u_{struc}(\bar{t}) = \max(\{\sigma\left(q_{ui}(\bar{t}) \cdot h^i(\bar{t})\right),\ \forall i \in N_u(\bar{t})\})$$

$q_{ui}(\bar{t})$ can be understood as the weight of the connection

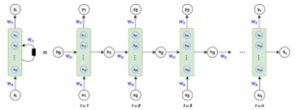
$$h^i(\bar{t}) = \mathbf{W}z^i(\bar{t}) + \boldsymbol{b}$$

Today

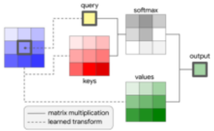
- Covered works for natural dynamics in GNNs
  - Temporal GNNs with Convolutional Operations



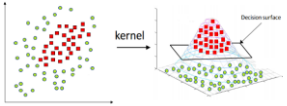
  - Temporal GNNs with Recurrent Units



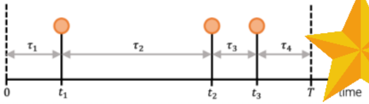
  - Temporal GNNs with Time Attention



  - Temporal GNNs with Time Kernel



  - Temporal GNNs with Temporal Point Process



There are also many great research works on natural dynamics

- Ma et al. **Streaming Graph Neural Networks**. SIGIR 2020
- Rossi et al. **Temporal Graph Networks for Deep Learning on Dynamic Graphs**. CoRR (2020)
- Tian et al. **Self-supervised Representation Learning on Dynamic Graphs**. CIKM 2021
- Fu et al. **SDG: A Simplified and Dynamic Graph Neural Network**. SIGIR 2021
- You et al. **ROLAND: Graph Learning Framework for Dynamic Graphs**. KDD 2022
- Cong et al. **Do We Really Need Complicated Model Architectures For Temporal Networks?** ICLR 2023
- Many more……

Q&A

Guest Lecture at CS6804 – Machine Learning on Graphs

Dongqi Fu
Ph.D. Candidate
Department of Computer Science
University of Illinois, Urbana-Champaign
dongqif2@illinois.edu
https://dongqifu.github.io/