Temporal Graph Representation Learning

Rakshit Trivedi
School of Computational Science and Engineering
rstrivedi@gatech.edu
Today’s Lecture

• GraphSAGE
• Dynamic Graphs and its Applications
• Representation Learning with:
  – Discrete-Time Approaches
  – Continuous-Time Approaches
GraphSAGE Idea

- In GCN, we aggregated the neighbors’ messages as the \((\text{weighted})\) average of all neighbors. How can we generalize this?

\[ \text{INPUT GRAPH} \]

\[ \text{TARGET NODE} \]

\[ \text{[Hamilton et al., NIPS 2017]} \]
Any differentiable function that maps set of vectors in $N(u)$ to a single vector

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

- **Simple neighborhood aggregation:**

\[
    h_{vk}^k = \sigma \left( W_k \sum_{u \in N(v)} \frac{h_{uv}^{k-1}}{|N(v)|} + B_k h_{vk}^{k-1} \right)
\]

- **GraphSAGE:**

**Concatenate neighbor embedding and self embedding**

\[
    h_{vk}^k = \sigma \left( [W_k \cdot \text{AGG} (\{h_{uv}^{k-1}, \forall u \in N(v)\})] , B_k h_{vk}^{k-1} \right)
\]

**Generalized aggregation**
Neighbor Aggregation: Variants

- **Mean:** Take a weighted average of neighbors
  \[
  \text{AGG} = \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|}
  \]

- **Pool:** Transform neighbor vectors and apply symmetric vector function
  \[
  \text{AGG} = \gamma \left( \{ Qh_u^{k-1}, \forall u \in N(v) \} \right)
  \]
  
  Element-wise mean/max

- **LSTM:** Apply LSTM to reshuffled of neighbors
  \[
  \text{AGG} = \text{LSTM} \left( [h_u^{k-1}, \forall u \in \pi(N(v))] \right)
  \]
Experiments: Dataset

• Dynamic datasets:
  – Citation Network: Predict paper category
    • Data from 2000-2005
    • 302,424 nodes
    • Train: data till 2004, test: 2005 data
  – Reddit Post Network: Predict subreddit of post
    • Nodes = posts
    • Edges between posts if common users comment on the post
    • 232,965 posts
    • Train: 20 days of data, test: next 10 days of data
## Experiments: Results

<table>
<thead>
<tr>
<th>Name</th>
<th>Citation Unsup. F1</th>
<th>Citation Sup. F1</th>
<th>Reddit Unsup. F1</th>
<th>Reddit Sup. F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.206</td>
<td>0.206</td>
<td>0.043</td>
<td>0.042</td>
</tr>
<tr>
<td>Raw features</td>
<td>0.575</td>
<td>0.575</td>
<td>0.585</td>
<td>0.585</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>0.565</td>
<td>0.565</td>
<td>0.324</td>
<td>0.324</td>
</tr>
<tr>
<td>DeepWalk + features</td>
<td>0.701</td>
<td>0.701</td>
<td>0.691</td>
<td>0.691</td>
</tr>
<tr>
<td>GraphSAGE-GCN</td>
<td>0.742</td>
<td>0.772</td>
<td><strong>0.908</strong></td>
<td>0.930</td>
</tr>
<tr>
<td>GraphSAGE-mean</td>
<td>0.778</td>
<td>0.820</td>
<td>0.897</td>
<td>0.950</td>
</tr>
<tr>
<td>GraphSAGE-LSTM</td>
<td>0.788</td>
<td>0.832</td>
<td><strong>0.907</strong></td>
<td><strong>0.954</strong></td>
</tr>
<tr>
<td>GraphSAGE-pool</td>
<td><strong>0.798</strong></td>
<td><strong>0.839</strong></td>
<td>0.892</td>
<td>0.948</td>
</tr>
<tr>
<td>% gain over feat.</td>
<td>39%</td>
<td>46%</td>
<td>55%</td>
<td>63%</td>
</tr>
</tbody>
</table>
Summary: GCN and GraphSAGE

- **Key idea:** Generate node embeddings based on local neighborhoods
  - Nodes aggregate “messages” from their neighbors using neural networks
- **Graph convolutional networks:**
  - **Basic variant:** Average neighborhood information and stack neural networks
- **GraphSAGE:**
  - Generalized neighborhood aggregation
Today’s Lecture

- GraphSAGE
- Dynamic Graphs and its Applications
- Representation Learning with:
  - Discrete-Time Approaches
  - Continuous-Time Approaches
Temporally Evolving Graphs

Dynamic Graphs are becoming Ubiquitous

- E-commerce
- Social media
- Web
- IoT
- Finance
- Education

Event Knowledge Graph

Temporal Social Network

Temporal Information Network
Temporally Evolving Graphs

(i) How to model dynamics over graphs?

(ii) How leverage such a dynamic graph model to encode evolving graph information into low-dimensional representations?
Application: Social Networks

D → S means S follows D

1:10pm, @D: Indeed
1:15pm, @S @D: Classic
1:20pm @C @S @D: Really? Will check
1:35pm @B @S @D: Indeed brilliant
2:03pm, @D: I want that car
1:30pm, @S @D: Very useful
2pm, @D: Nice car
1pm, @D: Cool paper
Application: Recommendation Systems

Users

Features

Time

Products

Srijan Kumar, Georgia Tech, CSE6240 Spring 2020: Web Search and Text Mining
Application: Anomaly Detection

[Image from NetWalk presentation, Yu et. al. KDD 2018]
How Do We Model Dynamics?

1. **Snapshot-Based Observation:**
   - Network Evolution observed as a collection of snapshots of the graph at different time steps.
   - Possibly significant changes in graph structure observed between the two-time steps.
   - Time information may or may not be explicitly available.
   - Demand Discrete-time modeling.

2. **Event Based Observation:**
   - Network Evolution observed as time-stamped edges (each edge represents an event).
   - Time information is fine-grained and explicitly available.
   - Demand Continuous-time modeling.
Today’s Lecture

- GraphSAGE
- Dynamic Graphs and its Applications
  - Representation Learning with:
    - Discrete-Time Approaches
    - Continuous-Time Approaches
Snapshot based Evolution of Graphs

- Let \( G_t = (V_t, E_t) \) denote the graph at time \( t \)
- Let \( A_t \) be the corresponding adjacency matrix at time \( t \)
- Dynamic graph \( G = \{G_1, G_2, \ldots, G_T\} \) is the series of graph snapshots recorded at \( T \) different time steps
Snapshot based Evolution of Graphs

- One Approach: Use a single graph encoder at each time step to extract node features
- Use RNN based model over these node features to model dynamics

What problems could this potentially have?
Snapshot based Evolution of Graphs

- Number of Nodes and edges vary with time step
- Above approach would require complete knowledge of nodes
- Doesn’t perform well in practice
Alternative Approach: Use graph-specific encoder at each time step
General Model

Alternative Approach: Use graph-specific encoder at each time step

- Adapt the architecture based on changes in graph properties
- Adapt Encoder parameters to model dynamics
- Train using unsupervised or semi-supervised loss as before e.g. cross-entropy loss
Variant I: Dynamic Autoencoder Architecture

DynGEM: Deep Embedding Method for Dynamic Graphs

[Slides for DynGEM adapted from author’s original slides, Goyal et. al. 2018]
DynGEM: Model
DynGEM: Adaptive Architecture

- Addition of nodes in the graph may require additional model parameters
- Get width hidden layers using PropSize heuristic
  \[ \text{size}(l_{k+1}) \geq \rho \times \text{size}(l_k) \]
- Deepen the model if PropSize is not satisfied for embedding layer
- Adopt Net2WiderNet and Net2DeeperNet to expand the autoencoder

\[ S_{rel}(F; t) = \frac{\|F_{t+1}(V_t) - F_t(V_t)\|_F}{\|F_t(V_t)\|_F} \frac{\|S_{t+1}(V_t) - S_t(V_t)\|_F}{\|S_t(V_t)\|_F} \]

- Relative change in embedding
- Relative change in graph

\[ K_S(F) = \max_{\tau, \tau'} |S_{rel}(F; \tau) - S_{rel}(F; \tau')| \]
DynGEM: Data Setup

- **Synthetic Data (SYN)**
  - Generated using Stochastic Block Model
  - 1000 nodes, 79,800-79,910 edges

- **High Energy Physics (HEP-TH)**
  - Author collaboration network
  - 1,424-7,980 nodes, 2,556-21,036 edges

- **Autonomous Systems (AS)**
  - Router communication network
  - 7716 nodes, 10,695-26,46 edges

- **Enron (ENRON)**
  - Email network
  - 184 nodes, 63-591 edges
DynGEM: Visualization

(a) DynGEM time step with 5 nodes jumping out of 1000

(b) DynGEM time step with 300 nodes jumping out of 1000
DynGEM: Link Prediction

- Randomly hide 15% of network edges at time $t$
- Train the model using graph snapshots till time $t$
- Test the prediction using hidden edges

<table>
<thead>
<tr>
<th>Model</th>
<th>SYN</th>
<th>HEP-TH</th>
<th>AS</th>
<th>ENRON</th>
</tr>
</thead>
<tbody>
<tr>
<td>$GF_{align}$</td>
<td>0.027</td>
<td>0.04</td>
<td>0.09</td>
<td>0.021</td>
</tr>
<tr>
<td>$GF_{init}$</td>
<td>0.024</td>
<td>0.042</td>
<td>0.08</td>
<td>0.017</td>
</tr>
<tr>
<td>SDNE$_{align}$</td>
<td>0.031</td>
<td>0.17</td>
<td>0.1</td>
<td>0.06</td>
</tr>
<tr>
<td>SDNE</td>
<td>0.034</td>
<td>0.1</td>
<td>0.09</td>
<td>0.081</td>
</tr>
<tr>
<td>DynGEM</td>
<td><strong>0.194</strong></td>
<td><strong>0.26</strong></td>
<td><strong>0.21</strong></td>
<td><strong>0.084</strong></td>
</tr>
</tbody>
</table>

**Table:** Average MAP of link prediction.
DynGEM: Anomaly Detection

- Detected anomalies in Enron by thresholding norm of change in consecutive embeddings
Variant II:
GCN Weight Evolution

EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs
EvolveGCN: Model

Node embedding

GCN 1
Layer 2 weights
Layer 1 weights

RNN 1

GCN 2
Layer 2 weights
Layer 1 weights

RNN 2

GCN 3
Layer 2 weights
Layer 1 weights

RNN 2

Time 1

Time 2

Time 3
EvolveGCN: Weight Evolution

• GCN Reminder:

\[ H^{(l+1)} = \sigma \left( H^{(l)}W^{(l)}_0 + \tilde{A}H^{(l)}W^{(l)}_1 \right) \]

\[ H^{(l+1)}_t = \text{GCONV}(A_t, H^{(l)}_t, W^{(l)}_t) \]

\[ := \sigma(\tilde{A}_tH^{(l)}_tW^{(l)}_t), \]
EvolveGCN: Weight Evolution

- **GCN Reminder:**

\[ H^{(l+1)} = \sigma \left( H^{(l)} W_0^{(l)} + \tilde{A} H^{(l)} W_1^{(l)} \right) \]

- **Weight Evolution I:**

(only structural properties)

- **Weight Evolution II:**

(for attributed graphs)

\[ H_t^{(l+1)} = \text{GCONV}(A_t, H_t^{(l)}, W_t^{(l)}) \]
\[ := \sigma(\tilde{A}_t H_t^{(l)} W_t^{(l)}) \]

\[ W_t^{(l)} = \text{LSTM}(W_{t-1}^{(l)}) \]

\[ W_t^{(l)} = \text{GRU}(H_t^{(l)}, W_{t-1}^{(l)}) \]
EvolveGCN: Weight Evolution

- **GCN Reminder:**

  \[ H^{(l+1)} = \sigma \left( H^{(l)} W_0^{(l)} + \tilde{A} H^{(l)} W_1^{(l)} \right) \]

  \[ H_t^{(l+1)} = \text{GCONV} (A_t, H_t^{(l)}, W_t^{(l)}) \]

  \[ := \sigma (\tilde{A}_t H_t^{(l)} W_t^{(l)}) \]

- **Weight Evolution I:**

  (only structural properties)

- **Weight Evolution II:**

  (for attributed graphs)
EvolveGCN: Summarization

What is the challenge?

\[
\begin{align*}
W_t^{(l)} & = \text{GRU}(H_t^{(l)}, W_{t-1}^{(l)}) \\
& \text{GCN weights} \quad \text{node embeddings} \quad \text{GCN weights}
\end{align*}
\]
EvolveGCN: Summarization

What is the challenge?
(Need to account for changing dimension of $H$)
EvolveGCN: Summarization

What is the challenge? (Need to account for changing dimension of H)

Use representative summarization:

```
function Z_t = summarize(X_t, k)
    y_t = X_t p / \| p \|
    i_t = \text{top-indices}(y_t, k)
    Z_t = [X_t \circ \tanh(y_t)]_{i_t}
end function
```

\[ W_t^{(l)} = \text{GRU}(H_t^{(l)}, W_{t-1}^{(l)}) \]
\[ := g(\text{summarize}(H_t^{(l)}, \#\text{col}(W_{t-1}^{(l)}))^T, W_{t-1}^{(l)}) \]
## EvolveGCN: Datasets

<table>
<thead>
<tr>
<th></th>
<th># Nodes</th>
<th># Edges</th>
<th># Time Steps (Train / Val / Test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBM</td>
<td>1,000</td>
<td>4,870,863</td>
<td>35 / 5 / 10</td>
</tr>
<tr>
<td>BC-OTC</td>
<td>5,881</td>
<td>35,588</td>
<td>95 / 14 / 28</td>
</tr>
<tr>
<td>BC-Alpha</td>
<td>3,777</td>
<td>24,173</td>
<td>95 / 13 / 28</td>
</tr>
<tr>
<td>UCI</td>
<td>1,899</td>
<td>59,835</td>
<td>62 / 9 / 17</td>
</tr>
<tr>
<td>AS</td>
<td>6,474</td>
<td>13,895</td>
<td>70 / 10 / 20</td>
</tr>
<tr>
<td>Reddit</td>
<td>55,863</td>
<td>858,490</td>
<td>122 / 18 / 34</td>
</tr>
<tr>
<td>Elliptic</td>
<td>203,769</td>
<td>234,355</td>
<td>31 / 5 / 13</td>
</tr>
</tbody>
</table>

- **SBM** *(Stochastic Block Model)* – Popular Model for simulating communities
- **BC-OTC; BC-Alpha**: who-trusts-whom network of Bitcoin users
- **UCI**: Messages sent between users in UC Irvine student community
EvolveGCN: Datasets

<table>
<thead>
<tr>
<th></th>
<th># Nodes</th>
<th># Edges</th>
<th># Time Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>(Train / Val / Test)</td>
</tr>
<tr>
<td>SBM</td>
<td>1,000</td>
<td>4,870,863</td>
<td>35 / 5 / 10</td>
</tr>
<tr>
<td>BC-OTC</td>
<td>5,881</td>
<td>35,588</td>
<td>95 / 14 / 28</td>
</tr>
<tr>
<td>BC-Alpha</td>
<td>3,777</td>
<td>24,173</td>
<td>95 / 13 / 28</td>
</tr>
<tr>
<td>UCI</td>
<td>1,899</td>
<td>59,835</td>
<td>62 / 9 / 17</td>
</tr>
<tr>
<td>AS</td>
<td>6,474</td>
<td>13,895</td>
<td>70 / 10 / 20</td>
</tr>
<tr>
<td>Reddit</td>
<td>55,863</td>
<td>858,490</td>
<td>122 / 18 / 34</td>
</tr>
<tr>
<td>Elliptic</td>
<td>203,769</td>
<td>234,355</td>
<td>31 / 5 / 13</td>
</tr>
</tbody>
</table>

- **AS (Autonomous Systems):** Communication network of routers that exchange traffic flows with peers
- **Reddit:** subreddit-to-subreddit hyperlink network, where each hyperlink originates from a post in the source community and links to a post in the target community
- **Elliptic:** bitcoin transactions, wherein each node represents one transaction and the edges indicate payment flows
EvolveGCN: Tasks

- Training performed end-to-end based on task

1. **Link Prediction:**

   For a pair of nodes $u$ and $v$, concatenate their embedding and apply an MLP to compute link probability

2. **Edge Classification:**

   For an edge $(u, v)$, similarly concatenate the corresponding node embedding and apply an MLP to compute edge class probability

3. **Node Classification:**

   For a node $u$, follow standard practice of using a softmax activation as the last layer of the GCN, thus outputting node class probability
EvolveGCN: Experiments

**Link Prediction**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>0.1987</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0251</td>
<td>0.0003</td>
<td>0.0138</td>
<td>0.0025</td>
<td>0.0031</td>
<td>0.1141</td>
<td>0.0555</td>
</tr>
<tr>
<td>GCN-GRU</td>
<td>0.1898</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0114</td>
<td>0.0713</td>
<td>0.0119</td>
<td>0.0003</td>
<td>0.0004</td>
<td>0.0985</td>
<td>0.3388</td>
</tr>
<tr>
<td>DynGEM</td>
<td>0.1680</td>
<td>0.0134</td>
<td>0.0525</td>
<td>0.0209</td>
<td>0.0529</td>
<td>0.0139</td>
<td>0.0921</td>
<td>0.1287</td>
<td>0.1055</td>
<td>0.1028</td>
</tr>
<tr>
<td>dyngraph2vecAE</td>
<td>0.0983</td>
<td>0.0090</td>
<td>0.0507</td>
<td>0.0044</td>
<td>0.0331</td>
<td>0.0079</td>
<td>0.0916</td>
<td>0.1478</td>
<td>0.0540</td>
<td>0.0698</td>
</tr>
<tr>
<td>dyngraph2vecAERNN</td>
<td>0.1593</td>
<td><strong>0.0220</strong></td>
<td><strong>0.1100</strong></td>
<td>0.0205</td>
<td>0.0711</td>
<td>0.0120</td>
<td><strong>0.1268</strong></td>
<td><strong>0.1945</strong></td>
<td>0.0713</td>
<td>0.0493</td>
</tr>
<tr>
<td>EvolveGCN-H</td>
<td>0.1947</td>
<td>0.0026</td>
<td>0.0049</td>
<td>0.0126</td>
<td><strong>0.1534</strong></td>
<td><strong>0.0141</strong></td>
<td>0.0690</td>
<td>0.1104</td>
<td>0.0899</td>
<td><strong>0.3632</strong></td>
</tr>
<tr>
<td>EvolveGCN-O</td>
<td><strong>0.1989</strong></td>
<td>0.0028</td>
<td>0.0036</td>
<td><strong>0.0270</strong></td>
<td>0.1139</td>
<td>0.0138</td>
<td>0.0968</td>
<td>0.1185</td>
<td><strong>0.1379</strong></td>
<td>0.2746</td>
</tr>
</tbody>
</table>

**Edge Classification**

**Node Classification**

Srijan Kumar, Georgia Tech, CSE6240 Spring 2020: Web Search and Text Mining
Summary So Far

- **GCN** → **GraphSAGE**

- **Next step:** **Dynamic graphs (time dimension)**
  - Applications: Social Media, Citation Network Analysis, Financial Transactions, Anomaly Detection and many more

- **Discrete Time Models for Snapshot Based Observation:**
  - Adaptive Architecture using Autoencoders
  - Adaptive Parameters using GCN and RNN

*(Does not make use of time information explicitly and cannot handle fine-grained complex temporal dynamics)*
Today’s Lecture

- GraphSAGE
- Dynamic Graphs and its Applications
  - Representation Learning with:
    - Discrete-Time Approaches
    - Continuous-Time Approaches
Event based Evolution of Graphs
Event based Evolution of Graphs

- \( G_t = (V_t, \mathcal{E}_t) \) denote graph \( G \) at time \( t \)

**Event Observation** – Dynamics are realized in the form of dyadic events observed between nodes on graph \( G \) over a temporal window \( [t_0, T] \) and ordered by time.

- \( e = (u, v, t, f) \) : Event at time \( t \), where \( u, v \) are the two nodes involved in an event. \( t \) represents time of the event. \( f \) can represent features associated with the event or any other model-specific quantity.

- Complete set of \( P \) observed events ordered by time in window \( [0, T] \) as \( \mathcal{O} = \{(u, v, t, f)_{p}\}_{p=1}^{P} \). Here, \( t_p \in \mathbb{R}^+ \), \( 0 \leq t_p \leq T \).

**Dynamic Graph Observations**

- Stream of events

- New nodes are always observed as part of events

- Displays Network Growth (Addition of Nodes and Edges)
Preliminaries: Graph Attention

**Simple neighborhood aggregation (GCN):**

\[
    h^k_v = \sigma \left( W^k \sum_{u \in N(v)} \frac{h^{k-1}_u}{|N(v)|} + B^k h^{k-1}_v \right)
\]

- Aggregates messages across neighborhood, \( N(v) \)
- \( q_{uv} = \frac{1}{|N(v)|} \) assigns weight (importance) of node \( u \)'s message to node \( v \)
- Explicitly based on structural properties of graph with each neighbor assigned equal importance
Preliminaries: Graph Attention

- Simple neighborhood aggregation (GCN):

\[
\mathbf{h}_v^k = \sigma \left( \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} \mathbf{W}_k + \mathbf{B}_k \mathbf{h}_v^{k-1} \right)
\]

- Aggregates messages across neighborhood, \( N(v) \)
- \( q_{uv} = \frac{1}{|N(v)|} \) assigns weight (importance) of node u’s message to node v
- Explicitly based on structural properties of graph with each neighbor assigned equal importance

Not all neighbors are equally important!
Preliminaries: Graph Attention

Can we learn the weight factors $q_{uv}$ implicitly?

Assign arbitrary importance to different neighbors of a node in the graph
Preliminaries: Graph Attention

Can we learn the weight factors $q_{uv}$ implicitly?

Assign arbitrary importance to different neighbors of a node in the graph

• **Attention Mechanism:**
  - While computing representation, nodes attend over their neighborhood
  - Implicitly specifying different weights to different nodes in a neighborhood
Preliminaries: Graph Attention Example

- **Key Idea:** Compute the hidden representations of each node in the graph, by attending over its neighbors, using a self-attention strategy.

![Graph Attention Example Diagram]

**Attended Aggregation:**

\[ z_u = \sum_{j \in N_u} q_{uj} z_j \]

**Example to compute q:**

[Velickovic et. al. 2018]

\[ q_{ui} = \frac{\exp(m_{ui})}{\sum_{i' \in N_u} \exp(m_{ui'})} \]

where, \( m_{ui} \) can be function of representations of nodes that form the edge \((u, i)\).
**Key Idea:** Compute the hidden representations of each node in the graph, by attending over its neighbors, using a self-attention strategy.

Normalize using **Softmax** to make it comparable across different neighborhoods.

$$m_{uv} = \alpha(W^k h_u^{k-1}, W^k h_v^{k-1})$$

\(\alpha\) – attention mechanism; could be single layer NN

**Attended Aggregation:**

$$z_u = \sum_{j \in N_u} q_{uj} z_j$$

**Example to compute** \(q\):

[Velickovic et. al. 2018]

$$q_{ui} = \frac{\exp(m_{ui})}{\sum_{i' \in N_u} \exp(m_{ui'})}$$

where, \(m_{ui}\) can be function of representations of nodes that form the edge \((u, i)\)
**Key Idea:** Compute the hidden representations of each node in the graph, by attending over its neighbors, using a self-attention strategy.

Normalize using Softmax to make it comparable across different neighborhoods.

**Multi-head attention to stabilize learning:**
- Repeat operation at each layer $R$ times
- Aggregate outputs

$$m_{uw} = a (W^k h_{u}^{k-1}, W^k h_{v}^{k-1})$$

$a$ – attention mechanism; could be single layer NN.

**Attended Aggregation:**

$$z^u = \sum_{j \in N_u} q_{uj} z_j$$

**Example to compute $q$:**

$$q_{ui} = \frac{\exp(m_{ui})}{\sum_{i' \in N_u} \exp(m_{ui'})}$$

where, $m_{ui}$ can be function of representations of nodes that form the edge $(u, i)$.
Preliminaries: Graph Attention Properties

- **Inductive:**
  - Shared edge-wise mechanism that does not depend on graph structure

- **Localized:**
  - Only attends over local network neighborhood

- **Computationally Efficient:**
  - Computation of attentional coefficient can be parallelized across all edges of the graph
  - Aggregation can be parallelized across nodes

**Key Benefit:** Allows for implicitly specifying different important values $q_{uv}$ to different neighbors
Preliminaries: Temporal Point Process

- Let \((t_i)_{i \in \mathbb{N}^*}\) a sequence of non-negative random variables such that \(\forall i \in \mathbb{N}^*, t_i < t_{i+1}\). We call \((t_i)_{i \in \mathbb{N}^*}\) a temporal point process on \(\mathbb{R}_+\).

- The variable \(t_i\) can represent the times of happening of events such as making posts, re-shares, likes, or comments.

- Let \((t_i)_{i \in \mathbb{N}^*}\) be a point process. The right-continues process \(N(t) = \sum_{i \in \mathbb{N}^*} 1_{t_i \leq t}\) is the counting process associated with \((t_i)_{i \in \mathbb{N}^*}\).

- Given \(\mathcal{H}_t\) is the history of all events up to time \(t\) the distribution of all events is given by the joint density

\[
f(t_1, t_2, \ldots) = \prod_i f(t_i \mid t_1, \ldots, t_{i-1}) = \prod_i f(t_i \mid \mathcal{H}_{t_i}) = \prod_i f^*(t_i)
\]
Preliminaries: Temporal Point Process

The *conditional intensity function* or *hazard function* is a convenient and intuitive way of specifying how the present depends on the past:

\[
\lambda^*(t) = \lim_{\Delta t \to 0} \frac{\mathbb{E}[N(t + \Delta t) - N(t) | \mathcal{H}_t]}{\Delta t}
\]

It represents the expected instantaneous rate of future events at time \( t \). The functional form of the intensity \( \lambda^*(t) \) is often designed to capture the phenomena of interests.

a) Poisson process

\[
\lambda^*(t) = \mu
\]

b) Hawkes process

\[
\lambda^*(t) = \mu + \alpha \sum_{t_i < t} \exp(-|t - t_i|) = \mu + \alpha \kappa_\omega(t) \ast dN(t)
\]

c) Survival process

\[
\lambda^*(t) = (1 - N(t)) g(t)
\]
Preliminaries: Temporal Point Process

- Specific form of Point Processes often suffer from model misspecification

- **Alternative:** Conditional Intensity function is parameterized by a Neural Network (often an RNN)

- Examples:
  - Recurrent Marked Temporal Point Process [Du et. al., 2016]
    \[
    \lambda^*(t) = \exp \left( \mathbf{v}^T \cdot \mathbf{h}_j + \mathbf{w}^t(t - t_j) + b^t \right)
    \]
  - Neural Hawkes Process [Mei et. al., 2017]
    \[
    \lambda_k(t) = f_k(\mathbf{w}_k^T \mathbf{h}(t))
    \]
    where,
    \[
    \mathbf{h}(t) = \mathbf{o}_i \odot (2\sigma(2\mathbf{c}(t)) - 1) \quad \text{for } t \in [t_{i-1}, t_i]
    \]
    where, \(\odot\) is an element-wise multiplication
Event Based Modeling of Complex Temporal Dynamics

DyRep: Representation Learning over Dynamic Graphs
Event Based Model

- [Chazelle et. al., 2012] The ability to express a dynamical process at different scales is an important feature of any influence system.
- Many dynamic graphs exhibit at least two processes that can be observed:
  - **Topological Evolution** (creates persistent edges; topology changes):
    - Initial Network
    - Jacob joins Network at 09:55 AM by befriending Bob
    - Jacob befriends Ann at 10:30 PM
  - **Network Interactions** (fixed topology; interacting nodes may be connected or non-connected):
    - Sophie interacts with Olivia at 09:00 AM
    - Sophie interacts with Bob at 10:00 AM
    - Jacob interacts with Ann at 08:00 PM
Evolution Through Mediation

Representation Learning as **Latent Mediation Process**
Dynamic of graph \(\iff\) **change of node's rep.** \(\iff\) Dynamic on graph

---

(c) Evolution Through Mediation through Embedding

Mutual Evolution through Embedding

(a) Dynamic of graph

Communication evolves Node Representations

Evolving Representations drive Communication

Association evolves Node Representation

Evolving Representations drive Association

Srijan Kumar, Georgia Tech, CSE6240 Spring 2020: Web Search and Text Mining
Social Network Example

Graph showing the evolution of social network at different time points:

- **t0**: Initial state with connections between Sophie, Olivia, Ann, Bob, and Jacob.
- **t1**: Sophie and Olivia communicate at time t1, evoking their embeddings.
- **t2**: Association between Bob and Jacob at time t2, evoking Bob's association with its neighbors.
- **t3**: Ann and Jacob communicate at time t3, evoking Ann and Jacob's embeddings.
- **t4**: Association between Ann and Jacob at time t4, evoking Ann's association with its neighbors.

Each node's embedding representation changes with communication events.
DyRep Model

- \( \bar{t} \): time point just before the current time point \( t \)
- Occurrence of event \( p \) corresponding to dynamics \( k \):

\[
\lambda_{k}^{u,v}(t) = f_k(g_{k}^{u,v}(\bar{t}))
\]  

(1)

where,

\[
f_k(x) = \psi_k \star \log(1 + \exp(x/\psi_k))
\]  

(2)

dynamics
(scale parameter)

- Intensity function is parameterized by deep representation network:

\[
g_{k}^{u,v}(\bar{t}) = \omega_{k}^T \cdot [z_{u}(\bar{t}); z_{v}(\bar{t})]
\]  

(3)

- Node representations \( z_{u}(\bar{t}) \) and \( z_{v}(\bar{t}) \) are computed using recurrent architecture

\[ e = (u, v, t, k) \]
- \( k=0 \): topology
- \( k=1 \): interaction
DyRep Model

- **Node Representation Update Function**

\[
\mathbf{z}^v(t_p) = \sigma \left( \mathbf{W}^{\text{struct}} \mathbf{h}^u_{\text{struct}}(\bar{t}_p) + \mathbf{W}^{\text{rec}} \mathbf{z}^v(t_p) + \mathbf{W}^t(t_p - \bar{t}_p) \right),
\]

\[ (4) \]

- **Exogenous Drive**: Smooth drift of nodes features over time
- **Self-Propagation**: Node evolves in an embedded space w.r.t to its previous positions (induces recurrence)
- **Localized Embedding Propagation**: Temporary or Permanent Pathway for information propagation between nodes (Illustrated later)

- **Computing** \( \mathbf{h}^u_{\text{struct}} \) **using max-pooling aggregation**:

\[
\mathbf{h}^u_{\text{struct}}(\bar{t}) = \max \left( \left\{ \sigma \left( \left\{ \frac{q_{ui}(t)}{\mathbf{W}^h z^i(\bar{t}) + \mathbf{b}^h} \right\} \cdot \mathbf{h}^i(\bar{t}) \right) \right\}, \forall i \in \mathcal{N}_u(\bar{t}) \right)
\]

\[ (5) \]

where, \( \mathbf{h}^i(\bar{t}) = \mathbf{W}^h z^i(\bar{t}) + \mathbf{b}^h \) (Simple MLP)
Localized Embedding Propagation

- Node $u$: update with information from $h^v_{\text{struct}}$ (green flow); Node $v$: update with information from $h^u_{\text{struct}}$ (red flow).
- Interaction events lead to temporary pathway (e.g., meeting at a conference).
- Topological events lead to permanent pathway (e.g., becoming friends).
Difference in number of blue arrows signify difference in importance of each neighbor to node $u \ (\mathcal{N}_u = \{1, 2, 3\})$ and node $v \ (\mathcal{N}_v = 5, 6, 7)$ respectively.

**Temporal Point Process Self-Attention:**

$$h^u_{struct}(\bar{t}) = \max(\{\sigma(q_{ui}(\bar{t}) \ast h^i(\bar{t}))\})$$

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

where $i \in \mathcal{N}_u(\bar{t})$ is the node in neighborhood of node $u$.

$$q_{ui}(\bar{t}) = \frac{\exp(S_{ui}(\bar{t}))}{\sum_{i' \in \mathcal{N}_u(\bar{t})} \exp(S_{ui'}(\bar{t}))}$$
Training Procedure

Objective: For a set $\mathcal{O}$ of $P$ observed events, minimize the intensity based (negative) log likelihood:

$$
\mathcal{L} = - \sum_{p=1}^{P} \log \left( \lambda_p(t) \right) + \int_{0}^{T} \Lambda(\tau) d\tau
$$

- $\lambda_p(t) = \lambda_{k_p}^{u_p,v_p}(t)$: Intensity of event at time $t$ (non-breakable sequence due to long intertwined history dependence) — Train over global sequence applying Backpropagation Through Time (BPTT) in sliding window fashion.

- $\Lambda(\tau) = \sum_{u=1}^{n} \sum_{v=1}^{n} \sum_{k \in \{0,1\}} \lambda_{k}^{u,v}(\tau)$: total survival probability for events that do not happen (intractable) — Algorithm II - Estimate survival term using Monte Carlo trick.
Experiments: Setup

- **Datasets**
  - MIT Social Evolution Dataset: Communication (Proximity, Calls, SMS); Association (Close Friendships)
  - Github dataset: Communication (Star/Watch); Association(Follow)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Nodes</th>
<th>#Initial Associations</th>
<th>#Final Associations</th>
<th>#Communications</th>
<th>Clustering Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social Evolution</td>
<td>83</td>
<td>376</td>
<td>791</td>
<td>2016339</td>
<td>0.548</td>
</tr>
<tr>
<td>Github</td>
<td>12328</td>
<td>70640</td>
<td>166565</td>
<td>604649</td>
<td>0.087</td>
</tr>
</tbody>
</table>

- **Train-Test Split: Time based**

- **Evaluation Scheme**
  - Divide Test data into multiple time slots and report for each slot
  - Sliding window protocol for baselines
  - **Metric:** MAR (Mean Absolute Rank) & HITS@10 for Dynamic Link Prediction
  - **Metric:** MAE (Mean Absolute Error) for Time Prediction
Experiments: Tasks

• **Temporal Link Prediction**
  
  **Task:** Which is the most likely node $u$ that would undergo an event of type $k$ with a given node $v$ at time $t$?

  Conditional Density: 
  $$f_k^{u,v}(t) = \lambda_k^{u,v}(t) \cdot \exp\left(\int_{\bar{t}}^{t} \lambda(s)ds\right)$$

  Use the conditional density to find the most likely node

• **Event Time Prediction**

  **Task:** Given two nodes $u$ and $v$ at previous time $\bar{t}$ in an event of dynamics $k$, when is the next time point $t$ for this event to occur again?

  Next time point: 
  $$t = \int_{\bar{t}}^{\infty} \bar{t}f_k^{u,v}(\bar{t})d\bar{t}$$
Experiment I: Dynamic Link Prediction

Communication Events (Top: MAR and Bottom: HITS@10)

[Graphs showing MAR and HITS@10 over time slots for Social Dataset and Github Dataset, with different methods such as DynGem, DyRep, Know-Evolve, DynTrd, GraphSage, and Node2Vec]
Experiment I: Dynamic Link Prediction

Association Events (Top: MAR and Bottom: HITS@10)

Social Dataset

Github Dataset
Experiment II: Event Time Prediction

- Mean Absolute Error (Top: Communication and Bottom: Association)

![Graphs showing MAE for different methods across time slots for Social and Github datasets.](image-url)
Today’s Lecture

• GraphSAGE
• Dynamic Graphs and its Applications
  • Representation Learning with:
    – Discrete-Time Approaches
    – Continuous-Time Approaches