Streaming GNN via Continual Learning [1]

Sicheng Mao, Yunhao Chen, Yang Zhang

MAP670G - Data Stream Processing

January 16, 2023

[1] original paper: Streaming Graph Neural Networks via Continual Learning

Overview

Framework: StreamingGNN

- In real case, network data is formed in a *streaming* fashion
  - Nodes and edges are modified over time.
- Patterns: Neighborhood information of nodes.
  - *New patterns* may appear and *Existing patterns* still maintain

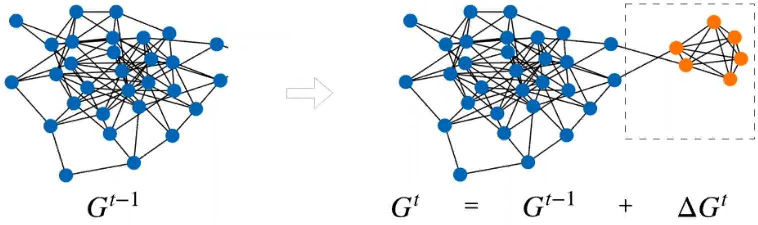
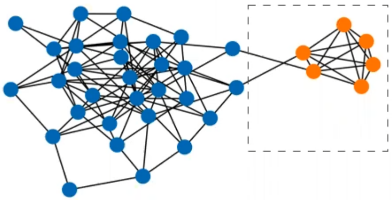


Figure: streaming networks.

StreamingGNN - current methods

- PretrainedGNN
  - Use a pre-trained GNN model to generate representations of unseen nodes and changed nodes on G^t : *Inductive learning*
  - Performance degrades if nodes are not in the pre-trained model.
- RetrainedGNN
  - Retrain the GNN at every time step.
  - High performance but computationally costly.



$$G^t = G^{t-1} + \Delta G^t$$

Figure: streaming networks.

StreamingGNN - current methods

- OnlineGNN
  - Train GNN based on ΔG^t using θ^{t-1} to initialize.
  - **Catastrophic forgetting** when the patterns in ΔG^t are different from θ^{t-1}, knowledge of θ^{t-1} may be abruptly lost.

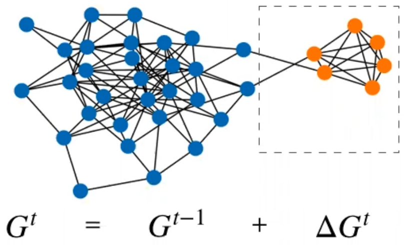
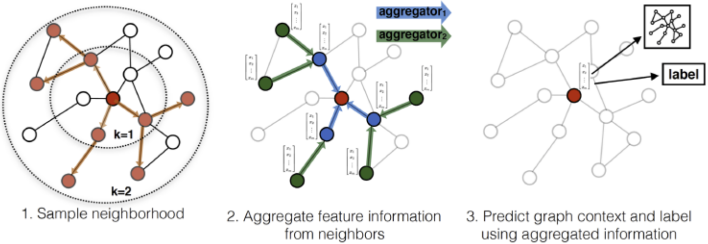


Figure: streaming networks.

Proposed method: ContinualGNN

- Problem to solve:
  Given streaming network G, at each time t, based on G^t the current graph. Detect new patterns incrementally while *preserving* existing patterns.

ContinualGNN - GraphSAGE



[2]

Figure: Visual illustration of the GraphSAGE sample and aggregate approach.

[2] Inductive Representation Learning on Large Graphs

ContinualGNN

- Two main tasks:
  - Current task: Training on ΔG^t
  - Previous task: Training on $(\Delta G^1, \Delta G^2, .., \Delta G^{t-1}) = G^{t-1}$

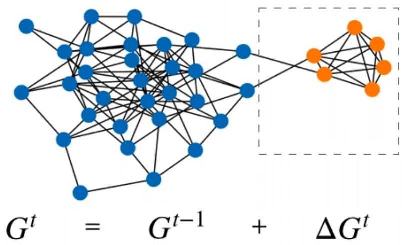


Figure: streaming networks.

ContinualGNN

- Loss function:
  - Using Bayes' rule:

    $$\log p(\theta|G^{t-1}, \Delta G^{t}) = \log p(\Delta G^{t}|\theta) + \log p(\theta|G^{t-1}) - \log p(\Delta G^{t})$$

  - General loss function of ContiunalGNN at time t is:

    $$L = L_{new} + L_{existing}$$

    where the first term L_{new} is the loss function on the influenced parts of networks, and $L_{existing} = L_{data} + L_{model}$ aims to consolidate patterns on previous data.

ContinualGNN

- General loss function of ContiunalGNN at time t is:

$$L = L_{new} + L_{data} + L_{model}$$

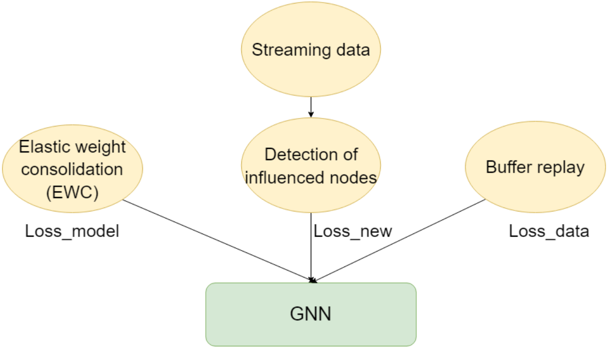


Figure: loss components.

Our contributions & Future

- Provide a more standard data format based on **Pytorch Geometric**
- **Refactor** the original code: more clean, efficient code
- **Remove the coupling relation** between the code with a specific dataset used in the paper

Problems

- Some problems of being integrated into (Deep) River:
  - ContinualGNN is not typically an online learning model, it has memories of the past data.
  - The ContinualGNN is based on graphSAGE, which is too complex and inefficient for (Deep) River.
  - The data type is different, for River the data should all be a dictionary, it's hard to convert graph data into a dictionary.