ROLAND

Framework for extending static GNNs to dynamic

Steven Morse
William & Mary, Data Science
DATA 691, Graph Learning

April 10, 2025

Overview

"ROLAND: Graph Learning Framework for Dynamic Graphs." Jiaxuan You, Tianyu Du, Jure Leskovec (Stanford), KDD 2022.

Idea

How can we adapt powerful GNN architectures directly to a dynamic setting?

Introduction and Related work

Adapting GNNs to dynamic setting introduces 3 challenges:

1. Model design. Sequence model on top of a GNN, RNN with GCN elements, ...
2. Evaluation setting. Partitioning temporal data risks distribution shifts.
3. Training strategy. Prohibitive memory overhead for dynamic training.

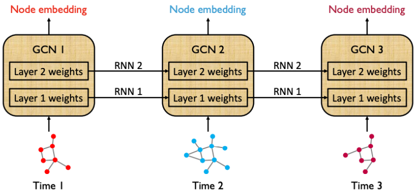
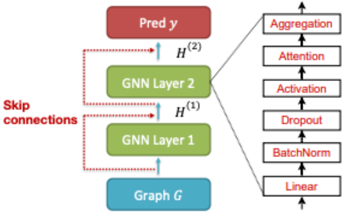


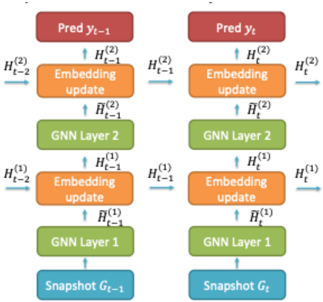
Figure: EvolveGCN

Extending static to dynamic out-of-the-box

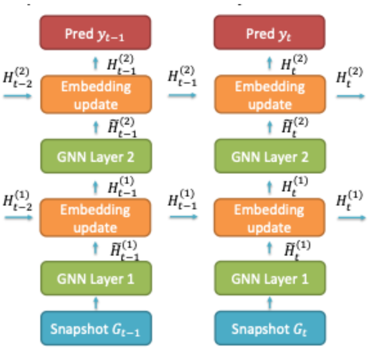
Idea

Instead of doing dynamic updates at the top-level, do them hierarchically at each layer.





A little more detail



Algorithm 1 ROLAND GNN forward computation

Input: Dynamic graph snapshot G_t, hierarchical node state H_{t-1}
Output: Prediction y_t, updated node state H_t

1: $H_t^{(0)} \leftarrow X_t$ {Initialize embedding from G_t}
2: **for** $l = 1, \ldots, L$ **do**
3: $\tilde{H}_t^{(l)} = \textsc{Gnn}^{(l)}(H_t^{(l-1)})$ {Implemented as Equation (4)}
4: $H_t^{(l)} = \textsc{Update}^{(l)}(H_{t-1}^{(l)}, \tilde{H}_t^{(l)})$ {Equation (2)}
5: $y_t = \textsc{Mlp}(\textsc{Concat}(\mathbf{h}_{u,t}^{(L)}, \mathbf{h}_{v,t}^{(L)})), \forall(u,v) \in E$ {Equation (5)}

"GNN" is:

$$\mathbf{m}_{u\to v}^{(l)} = \mathbf{W}^{(l)}\textsc{Concat}^{(l)}(\mathbf{h}_u^{(l-1)}, \mathbf{h}_v^{(l-1)}, \mathbf{f}_{uv})$$
$$\mathbf{h}_v^{(l)} = \textsc{Agg}^{(l)}(\{\mathbf{m}_{u\to v}^{(l)} | u \in N(v)\}) + \mathbf{h}_v^{(l-1)}$$

How to update?

- Moving average:
  $H_{t,v} = \kappa_{t,v} H^{(l)}_{t-1,v} + (1-\kappa_{t,v})\tilde{H}^{(l)}_{t,v}$.

$$\kappa_{t,v} = \frac{\sum_{\tau=1}^{t-1} |E_\tau|}{\sum_{\tau=1}^{t-1} |E_\tau| + |E_t|}$$

- MLP: $H^{(l)}_t = \mathrm{MLP}(\textsc{Concat}(H^{(l)}_{t-1}, \tilde{H}^{(l-1)}_t))$
- GRU: $H^{(l)}_t = \mathrm{GRU}(H^{(l)}_{t-1}, \tilde{H}^{(l-1)}_t)$

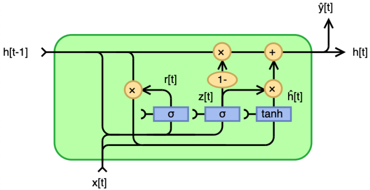


Figure: GRU

Wait — isn't this the same as EvolveGCN?

"While many works combine GNNs with recurrent models, we want to emphasize that our ROLAND framework is quite different."

EvolveGCN: weight params evolve (via RNN)

$$H_t^{(l+1)} = \textsc{Gnn}(H_t^{(l)}; W_t^{(l)})$$
$$W_t^{(l)} = \textsc{Update}(H_t^{(l)}, W_{t-1}^{(l)})$$

Memory efficient, but historical information slowly decays and we're limited to simpler GNN architectures.

ROLAND: weight params are static (w/in t)

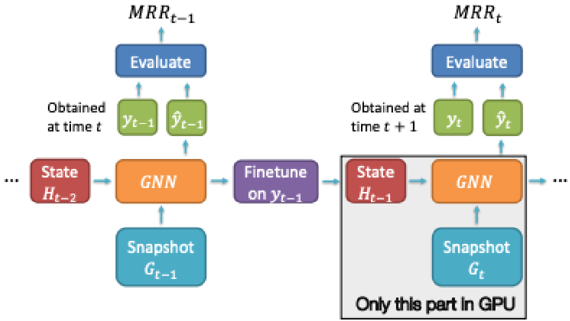
$$\tilde{H}_t^{(l+1)} = \textsc{Gnn}(H_t^{(l)}; W_t^{(l)})$$
$$H_t^{(l)} = \textsc{Update}(H_{t-1}^{(l)}, \tilde{H}_t^{(l)})$$

Not memory efficient (but fixed through their training strat), no decay, and no constraint on architecture.

Live-update evaluation and meta-training

Idea

Account for temporal distribution shifts by 1) training on the fly and 2) training a meta-model.



Algorithm 2 ROLAND live-update evaluation

Input: Dynamic graph $\mathcal{G} = \{G_1, \dots, G_T\}$, link prediction labels $y_1, \dots, y_T$, number of snapshots T, $\textsc{Gnn}(\cdot)$ defined in Algorithm 1
Output: Performance MRR, model $\textsc{Gnn}$

Initialize hierarchical node state H_0
for $t = 2, \dots, T$ **do**
  Collect link prediction labels $y_{t-1} = y_{t-1}^{(train)} \cup y_{t-1}^{(val)}, y_t$
  while $\text{MRR}_{t-1}^{(val)}$ is increasing **do**
    $H_{t-1}, \hat{y}_{t-1} \leftarrow \textsc{Gnn}(G_{t-1}, H_{t-2})$, $\hat{y}_{t-1} = \hat{y}_{t-1}^{(train)} \cup \hat{y}_{t-1}^{(val)}$
    Update $\textsc{Gnn}$ via backprop based on $\hat{y}_{t-1}^{(train)}, y_{t-1}^{(train)}$
    $\text{MRR}_{t-1}^{(val)} \leftarrow \textsc{Evaluate}(\hat{y}_{t-1}^{(val)}, y_{t-1}^{(val)})$
  $H_t, \hat{y}_t \leftarrow \textsc{Gnn}(G_t, H_{t-1})$
  $\text{MRR}_t \leftarrow \textsc{Evaluate}(\hat{y}_t, y_t)$
$\text{MRR} = \sum_{t=2}^{T} \text{MRR}_t / (T-1)$

Algorithm 3 ROLAND training algorithm

Input: Graph snapshot G_t, link prediction label y_t, hierarchical node state H_{t-1}, smoothing factor α, meta-model $\textsc{Gnn}^{(meta)}$
Output: Model $\textsc{Gnn}$, updated meta-model $\textsc{Gnn}^{(meta)}$

$\textsc{Gnn} \leftarrow \textsc{Gnn}^{(meta)}$
Move $\textsc{Gnn}, G_t, H_{t-1}$ to GPU
while $\text{MRR}_t^{(val)}$ is increasing **do**
  $H_t, \hat{y}_t \leftarrow \textsc{Gnn}(G_t, H_{t-1})$, $\hat{y}_t = \hat{y}_t^{(train)} \cup \hat{y}_t^{(val)}$
  Update $\textsc{Gnn}$ via backprop based on $\hat{y}_t^{(train)}, y_t^{(train)}$
  $\text{MRR}_t^{(val)} \leftarrow \textsc{Evaluate}(\hat{y}_t^{(val)}, y_t^{(val)})$
Remove $\textsc{Gnn}, G_t, H_{t-1}$ from GPU
$\textsc{Gnn}^{(meta)} \leftarrow (1-\alpha)\textsc{Gnn}^{(meta)} + \alpha\textsc{Gnn}$

Experiments

	Bitcoin-OTC	Bitcoin-Alpha	UCI-Message
GCN [14]	0.0025	0.0031	0.1141
DynGEM [12]	0.0921	0.1287	0.1055
dyngraph2vecAE [8]	0.0916	0.1478	0.0540
dyngraph2vecAERNN [8]	**0.1268**	**0.1945**	0.0713
EvolveGCN-H [30]	0.0690	0.1104	0.0899
EvolveGCN-O [30]	0.0968	0.1185	**0.1379**
ROLAND Moving Average	0.0468 ± 0.0022	0.1399 ± 0.0107	0.0649 ± 0.0049
ROLAND MLP	0.0778 ± 0.0024	0.1561 ± 0.0114	0.0875 ± 0.0110
ROLAND GRU	**0.2203 ± 0.0167**	**0.2885 ± 0.0123**	**0.2289 ± 0.0618**
Improvement over best baseline	73.74%	43.33%	65.99%

Figure: Standard fixed-split setting

	BSI-ZK	AS-733	Reddit-Title	Reddit-Body	BSI-SVT	UCI-Message	Bitcoin-OTC	Bitcoin-Alpha
	MRR	MRR	MRR	MRR	MRR	MRR	MRR	MRR
Baseline Models with standard training								
EvolveGCN-H	N/A, OOM	N/A, OOM	N/A, OOM	**0.148 ± 0.013**	**0.031 ± 0.016**	0.061 ± 0.040	0.067 ± 0.035	0.079 ± 0.032
EvolveGCN-O	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	0.015 ± 0.006	0.071 ± 0.009	0.085 ± 0.022	0.071 ± 0.025
GCRN-GRU	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	0.080 ± 0.012	N/A, OOM	N/A, OOM
GCRN-LSTM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	**0.083 ± 0.001**	N/A, OOM	N/A, OOM
GCRN-Baseline	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	0.069 ± 0.004	**0.152 ± 0.011**	**0.141 ± 0.005**
TGCN	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	0.054 ± 0.024	0.128 ± 0.049	0.088 ± 0.038
Baseline Models with ROLAND Training								
EvolveGCN-H	N/A, OOM	0.251 ± 0.079	0.165 ± 0.026	0.102 ± 0.010	0.032 ± 0.008	0.057 ± 0.012	0.076 ± 0.022	0.054 ± 0.015
EvolveGCN-O	0.396	0.163 ± 0.002	0.047 ± 0.004	0.033 ± 0.001	0.018 ± 0.003	0.066 ± 0.012	0.032 ± 0.004	0.034 ± 0.002
GCRN-GRU	N/A, OOM	**0.344 ± 0.001**	0.338 ± 0.006	0.217 ± 0.004	0.050 ± 0.004	0.089 ± 0.004	0.173 ± 0.003	0.140 ± 0.004
GCRN-LSTM	N/A, OOM	0.341 ± 0.001	0.344 ± 0.005	0.216 ± 0.000	0.051 ± 0.002	0.091 ± 0.010	0.174 ± 0.004	**0.146 ± 0.005**
GCRN-Baseline	0.754	0.336 ± 0.002	0.351 ± 0.001	0.218 ± 0.002	0.054 ± 0.002	**0.095 ± 0.008**	**0.183 ± 0.002**	0.145 ± 0.003
TGCN	**0.831**	0.343 ± 0.002	**0.391 ± 0.004**	**0.251 ± 0.001**	**0.157 ± 0.004**	0.080 ± 0.015	0.083 ± 0.011	0.069 ± 0.013
ROLAND results								
Moving Average	0.819	0.309 ± 0.011	0.362 ± 0.007	0.289 ± 0.038	0.177 ± 0.006	0.075 ± 0.006	0.120 ± 0.002	0.0962 ± 0.010
MLP-Update	0.834	0.329 ± 0.021	0.395 ± 0.006	0.291 ± 0.008	**0.217 ± 0.003**	0.103 ± 0.010	0.154 ± 0.010	0.148 ± 0.012
GRU-Update	**0.851**	**0.340** ± 0.001	**0.425** ± 0.015	**0.362** ± 0.002	0.205 ± 0.014	**0.112** ± 0.008	**0.194** ± 0.004	**0.157** ± 0.007
Improvement over the best baseline	2.40%	-1.16%	8.70%	44.22%	38.21%	17.89%	6.01%	7.53%

Figure: Live update evaluation

Experiments

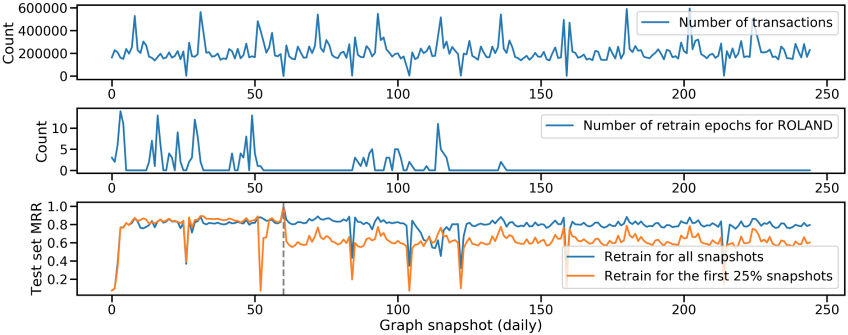


Figure: ROLAND model retraining

Subsequent and related work

- (Related) GraphGym framework (SNAP-Stanford)
- ROLAND = Discrete time, integrated. (See taxonomy)
- Extensions: WinGNN (2023)
- Applications: network biology (2024)

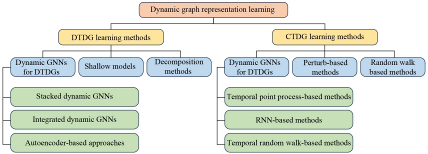


Figure: Dynamic graph learning taxonomy (from Zheng et al, 2025)

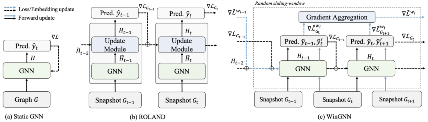


Figure: WinGNN (2023)

Questions?