Dada-DyGNN: Deletion-Augmented Streaming GNNs

APMTH 220 Poster Session, Harvard University, Cambridge, Massachusetts 02138

Jerry Liang

Abstract

Graph neural networks (GNNs) are a formidable tool in learning useful representations for node and edge features. However, most existing work focus on static graphs, ignoring the complex dynamic nature of most real-world applications. Existing dynamic state-of-the-art methods only consider interactions in the form of edge insertions and neglect edge deletions. In Dada-DyGNN, we modify existing work in Ada-DyGNN to incorporate edge deletions with temporal decay in reinforcement learning context. Our work differs from existing literature since we do not want to explicitly "unlearn" that an interaction ever occurred–rather we wish to leverage all past history to learn a rich embedding useful for the downstream task of link-prediction. Our contributions are twofold: (1) a theoretical redesign of the model architecture and reward scheme; and (2) open-source implementation with deletion-augmentation and patches of existing bugs in Ada-DyGNN. Results are limited due to our compute constraints, but show promise given hyperparameter adjustments and ample resources in converging to a final neighbor update policy.

Introduction and Related Work

Dynamic GNNs (DyGNNs), first pioneered in 1997, were the first graph neural networks that allowed for streamed edges. Since then, many variants of DyGNNs have been developed—continuous time networks, autoencoder models (DynGEM), random walks (JODIE), temporal graph attention, temporal graph networks, and reinforcement learning augmented methods (Ada-DyGNN). We extend Ada-DyGNN with **D**elete-Augmented **Ada**ptive **Dy**namic **GNN** (Dada-DyGNN).

Edge deletions in graph streaming and link-prediction have been sparsely studied. The most pertinent publications in this area are GNNDelete and Decoupled GNNs. GNNDelete focuses on the task of "*unlearning*", which explicitly leave node representations as if the deleted edge was *never added* in the first place. Decoupled GNNs are closer to our work, but deletions in this decoupled scheme correspond to purely negating a weight update, offering not much sophistication besides partial unlearning. Likewise, this paper focuses on the traditional insertions-only link-prediction. In Dada-DyGNN, we incorporate nuanced temporal update information (rather than direct unlearning) in specifically an RL context, and test harder downstream tasks of both insert & delete link-predictions.

Background and Notation

Dada-DyGNN is built on top of the existing Ada-DyGNN model architecture. There are three core parts of this existing RL scheme of Robust Knowledge Adaptation: (1) the *Time-Aware Attentional Aggregating Module* (hereby T3AM); (2) the *Reinforced Neighbor Selection Module* (hereby RNSM); and (3) *Downstream Link-Prediction*. In brief, T3AM incorporates temporal data into the message passing paradigm; RNSM uses these temporal-augmented intermediate states to choose which neighbors to retain versus update; and link-prediction is done after representations are learned through cosine similarity (with highly similar nodes likelier to have an edge inserted at the next timestamp).

T3AM: Edge insertion (v_s, v_d, t) with historical interactions $(v_s, v_1, t_1), \ldots, (v_s, v_k, t_k)$. Computes intermediate states $\mathbf{h}_i(t)$ to update embeddings $\mathbf{x}_i(t)$. Define $\Delta t_i = t - t_i$ and time-decay $\phi(\Delta t) = \frac{1}{1+\Delta t}$. The time-aware attention coefficient between two vertices is as follows, σ is LeakyReLU, σ_1 is ReLU, and $\|$ represents concatenation.

$$\alpha_{si} = \frac{\exp\left(\sigma\left(\mathbf{a}^\top\left[\mathbf{W}_g\mathbf{x}_s(t) \,\|\, \phi(\Delta t_i)\mathbf{W}_g\mathbf{x}_i(t)\right]\right)\right)}{\sum_{j\in\mathcal{N}_s(t)}\exp\left(\sigma\left(\mathbf{a}^\top\left[\mathbf{W}_g\mathbf{x}_s(t) \,\|\, \phi(\Delta t_j)\mathbf{W}_g\mathbf{x}_j(t)\right]\right)\right)} \quad \mathbf{m}_s(t) = \sigma_1\left(\sum_{i\in\mathcal{N}_s(t)} \alpha_{si}\,\phi(\Delta t_i)\,\mathbf{W}_g\mathbf{x}_i(t)\right) \in \mathbb{R}^{d_m/2} \quad \mathbf{m}(t) = \mathbf{m}_s(t) \,\|\, \mathbf{m}_d(t) \,\|\, \sigma_1(\mathbf{W}_e\mathbf{e}(t)) \in \mathbb{R}^{2d_m}$$

which gives vertex & interaction messages as shown above. Finally, this defines the attentional coefficient of a node and the intermediate state passed to RNSM.

$$\beta_i = \frac{\exp\left(\sigma_2\left(\phi(\Delta t_i)\mathbf{x}_i(t)\cdot\mathbf{W}_p\mathbf{m}(t)\right)\right)}{\sum_{j\in\mathcal{N}_{s\cup d}(t)}\exp\left(\sigma_2\left(\phi(\Delta t_j)\mathbf{x}_j(t)\cdot\mathbf{W}_p\mathbf{m}(t)\right)\right)} \qquad \mathbf{h}_i(t) = \beta_i\mathbf{W}_p\mathbf{m}(t) \in \mathbb{R}^{d_n}$$

RNSM: State $\mathbf{s}_i(t) = \mathbf{h}_i(t) \,\|\, \mathbf{x}_i(t)$ with policy $\pi(\mathbf{s}_i(t)) = \sigma_2(\mathbf{W}_1\sigma_1(\mathbf{W}_2\mathbf{s}_i(t)))$ where σ_2 is sigmoid. Embeddings are updated via an action, and cosine similarity reward preserves local stability.

$$\mathbf{x}_i(t+) = \begin{cases}\sigma_1(\mathbf{W}_u(\mathbf{x}_i(t) \,\|\, \mathbf{h}_i(t)) & \text{if } a_i = 1,\\ \mathbf{x}_i(t) & \text{if } a_i = 0\end{cases} \qquad r = \frac{\sum_{i\in\mathcal{N}_s^+(t)}\cos\left(\mathbf{x}_s(t+),\mathbf{x}_i(t+)\right)}{|\mathcal{N}_s^+(t)|} + \frac{\sum_{i\in\mathcal{N}_d^+(t)}\cos\left(\mathbf{x}_d(t+),\mathbf{x}_i(t+)\right)}{|\mathcal{N}_d^+(t)|}$$

Policy parameters are updated with self-critical training, and T3AM uses cross-entropy loss from a randomly sampled negative edge.

$$\theta \leftarrow \theta + \eta\,\frac{1}{|\mathcal{N}_{s\cup d}(t)|}\sum_{i\in\mathcal{N}_{s\cup d}(t)}(r-\hat{r})\nabla_\theta\log\pi_\theta(\mathbf{s}_i(t)) \quad L_{ce} = -\log(\sigma_2(\mathbf{x}_s(t+)^\top\mathbf{x}_d(t+))) - \log(\sigma_2(1-\mathbf{x}_s(t+)^\top\mathbf{x}_n(t+)))$$

Link-Prediction: Link prediction utilizes cosine similarity $\cos(\mathbf{x}_i, \mathbf{x}_j)$and evaluated on Mean Reciprocal Rank $MRR = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{\text{rank}_i}$ as well as area under the curve/average precision (binary classification).

Approach

To support an insert-delete streaming paradigm, we propose significant, but natural, changes to both the Ada-DyGNN model architecture and the downstream link-prediction task. We modify inputs to be (v_s, v_d, t, f) where f is a flag that is 1 for insertions & –1 for deletions. We consider the most recent insertions/deletions (t/t'), but our method can be easily extended to a k-length history.

T3AM: We modify attentional coefficients to be softmax of $\sigma\left(\mathbf{a}^\top\left[\mathbf{W}_g\mathbf{x}_s(t) \,\|\, \phi(\Delta t_i)\mathbf{W}_g\mathbf{x}_i(t) \,\|\, \phi(\Delta t_i')(-\mathbf{W}_g)\mathbf{x}_i(t)\right]\right)$ and modify messages/matrix dimensions accordingly (see summary table).

$$\mathbf{m}_s(t) = \sigma_1\left(\sum_{i\in\mathcal{N}_s(t)} \alpha_{si}\,\left[\phi(\Delta t_i)\,\mathbf{W}_g\mathbf{x}_i(t) \,\|\, \phi(\Delta t_i')\,(-\mathbf{W}_g)\mathbf{x}_i(t)\right]\right) \in \mathbb{R}^{2d_m} \qquad \mathbf{m}(t) = \mathbf{m}_s(t) \,\|\, \mathbf{m}_d(t) \,\|\, \sigma_1(\mathbf{W}_e\mathbf{e}(t)) \in \mathbb{R}^{8d_m}$$

RNSM: We concatenate the flag f in $\mathbf{s}_i(t) = \mathbf{h}_i(t) \,\|\, \mathbf{x}_i(t) \,\|\, f_i$ and modify T3AM cross-entropy loss by properly sampling negative edges.

$$L_{ce} = \begin{cases}-\log(\sigma_2(\mathbf{x}_s(t+)^\top\mathbf{x}_d(t+))) - \log(\sigma_2(1-\mathbf{x}_s(t+)^\top\mathbf{x}_n(t+))) & \text{if } f=1, \text{ randomly sample } (v_s, v_n)\notin E,\\ -\log(1-\sigma_2(\mathbf{x}_s(t+)^\top\mathbf{x}_d(t+))) - \log(\sigma_2(\mathbf{x}_s(t+)^\top\mathbf{x}_n(t+))) & \text{if } f=-1, \text{ randomly sample } (v_s, v_n)\in E.\end{cases}$$

Link-Prediction: Instead of dot-product, we train an MLP acting on the entrywise product $\mathbf{x}_i \odot \mathbf{x}_j$, and interpret this value with tanh (-1: delete, +1: insert). We train this MLP separately with CE loss.

$$\text{MLP}(\mathbf{x}_i \odot \mathbf{x}_j) = \mathbf{W}_1^{\text{MLP}}\sigma_1(\mathbf{W}_2^{\text{MLP}}(\mathbf{x}_i \odot \mathbf{x}_j)) \qquad \text{Predict}(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\text{MLP}(\mathbf{x}_i \odot \mathbf{x}_j)) \in [-1, 1]$$

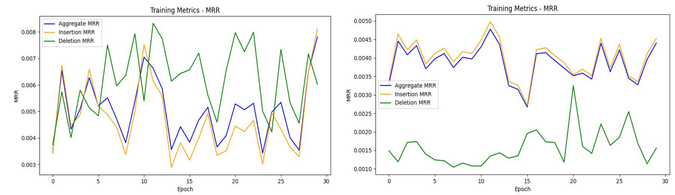
$$L_{ce}^{\text{MLP}} = \begin{cases}-\log(\sigma_2(\text{MLP}(\mathbf{x}_i \odot \mathbf{x}_j))) - \log(\sigma_2(1-\text{MLP}(\mathbf{x}_i \odot \mathbf{x}_k))) & \text{if } f=1, \text{ randomly sample } (v_i, v_k)\notin E,\\ -\log(1-\sigma_2(\text{MLP}(\mathbf{x}_i \odot \mathbf{x}_j))) - \log(\sigma_2(\text{MLP}(\mathbf{x}_i \odot \mathbf{x}_k))) & \text{if } f=-1, \text{ randomly sample } (v_i, v_k)\in E.\end{cases}$$

Training: Due to a (surprising) lack of publicly-available timestamped social network data with deletions, we benchmark Dada-DyGNN by testing future link prediction on two synthetic datasets. First, we augment the existing UCI dataset, deleting an inserted edge with probability p after T ~ Expo(λ) from its first interaction. We chose p = 0.2 and 1/λ = 0.01 × (T_{max} – T_{min}). Second, we generate a fully synthetic preferential-attachment model (edges inserted/deleted based on node degrees iteratively). Theoretical results show this converges to the power law distribution pervasive across real world phenomena. These timestamps are distributed as a Poisson process, 1/λ = 100.

Component	Symbol	Changes in Dada-DyGNN	Dimensions
Input	(v_s, v_d, t, f)	Add flag $f \in \{+1, -1\}$ corresponding to insert/delete	N/A
Time-Aware Attentional Coefficient	α_{si}	Concatenate edge deletion history $\phi(\Delta t_i')(-\mathbf{W}_g)\mathbf{x}_i(t)$ before softmax using shared weights $\mathbf{W}_g$	$\mathbf{W}_g \in \mathbb{R}^{d_m\times d_n}$ $\mathbf{a} \in \mathbb{R}^{3d_m}$
(Node) Messages	$\mathbf{m}_s(t), \mathbf{m}_d(t), \mathbf{m}(t)$	Concatenate edge deletion history $\phi(\Delta t_i')(-\mathbf{W}_g)\mathbf{x}_i(t)$	$\mathbf{m}_s(t), \mathbf{m}_d(t) \in \mathbb{R}^{2d_m}$ $\mathbf{W}_e \in \mathbb{R}^{4d_m\times d_e}$
(Intermediate) States	$\mathbf{h}_i(t), \mathbf{s}_i(t)$	No modification beyond weight matrix; Concatenate flag f_i	$\mathbf{W}_p \in \mathbb{R}^{d_n\times 8d_m}$ $\mathbf{s}_i(t) \in \mathbb{R}^{2d_n+1}$ $\mathbf{W}_2 \in \mathbb{R}^{d_n\times(2d_n+1)}$
T3AM Loss Function	L_{ce}	Different cases based on insert/delete; update negative sampling	N/A
Link-Prediction	N/A	Train MLP on $\mathbf{x}_i \odot \mathbf{x}_j$ with cross-entropy loss (σ_2); output "probability" of insert (+1)/delete (-1) as tanh	$\mathbf{W}_1^{\text{MLP}} \in \mathbb{R}^{1\times d_n}$ $\mathbf{W}_2^{\text{MLP}} \in \mathbb{R}^{d_n\times d_n}$

Results

For each set, we consider only MRR (strictly harder metric than AUC). We also use a similar hyperparameter setup to Ada-DyGNN. We use the Adam optimizer with 0.5 dropout rate, 0.0001 learning rate, and a maximum of 30 epochs (due to compute limitations). Our lightweight MLP classifier is trained over 1000 epochs. We set d_n = 64 and d_m = 16 consistently throughout the experiment. This corresponds to messages & embeddings of the same dimension as Ada-DyGNN for comparability. Furthermore, NSRM only samples the most k = 200 recent interaction neighbors. In both transductive/inductive settings, the first 80% of the edges are used as the training set, 10% are used as the validation set, and the remaining 10% as the testing set. Our experiments are run locally on a 8-core AMD Ryzen 7 5700U CPU with Radeon Graphics, taking 32660.34s (~ 9h) for our augmented UCI dataset (left), and 14037.29s (~ 4h) for our preferential attachment model (right).



Discussion

Dada-DyGNN achieves a MRR (transductive/inductive) of around 0.007/0.004 for UCI; 0.004/0.003, for PA. We attribute the comparatively poor performance to lack of convergence & the inherent difficulty of *simultaneously* predicting deletions & insertions. It's also possible that constraining messages to the same dimension greatly reduces the expressivity of our GNN. As expected, with a larger proportion of deletion updates, relative performance increases for deletion, e.g., in PA (5% deletions), we only achieve a MRR of roughly 0.002, but surprisingly, in UCI (20%), performance on deletion is better than insertion. This shows promise that link-deletion can be incorporated as a downstream task for dynamic GNNs. Lastly, the classifier MLP failed to improve performance, with MRRs of 0.0033/0.0040 for UCI; 0.0031/0.0027 for PA—lower than raw MRRs. We hypothesize this is due to overfitting over 1000 epochs.

Future Directions

Clear next steps include more ablations, in particular: training for more epochs to achieve true policy convergence; decreasing MLP training to prevent overfitting; plotting performance across message/embedding dimensions; increasing history length (beyond the latest deletion/insertion); and testing on non-augmented real-world data.

Beyond this, it would be worthwhile to explore how to extend more general classes of GNNs. In particular, how well does concatenation with time-decay and a look-back history work for general graph learning tasks? Can the loss function be improved upon with contrastive learning? Should time-decay be explicitly predetermined (e.g., in our case, O(1/Δt)) or implicitly learned?