Learning to Evolve on Dynamic Graphs (Student Abstract)

Xintao Xiang[1]*, **Tiancheng Huang**[2,3,4]*, **Donglin Wang**[3,4] †

[1] Australian National University, Canberra, Australia [2] Zhejiang University, Hangzhou, China
[3] Westlake University, Hangzhou, China [4] Westlake Institute for Advanced Study, Hangzhou, China
xintao.xiang@anu.edu.au, {huangtiancheng,wangdonglin}@westlake.edu.cn

Abstract

Representation learning in dynamic graphs is a challenging problem because the topology of graph and node features vary at different time. This requires the model to be able to effectively capture both graph topology information and temporal information. Most existing works are built on recurrent neural networks (RNNs), which are used to exact temporal information of dynamic graphs, and thus they inherit the same drawbacks of RNNs. In this paper, we propose Learning to Evolve on Dynamic Graphs (LEDG) - a novel algorithm that jointly learns graph information and time information. Specifically, our approach utilizes gradient-based meta-learning to learn updating strategies that have better generalization ability than RNN on snapshots. It is model-agnostic and thus can train any message passing based graph neural network (GNN) on dynamic graphs. To enhance the representation power, we disentangle the embeddings into time embeddings and graph intrinsic embeddings. We conduct experiments on various datasets and down-stream tasks, and the experimental results validate the effectiveness of our method.

Introduction

Representation learning on graph data has received increasing attention. However, most works focus on the static graph and ignore the fact that many real-world graphs are time-dependant. Our work falls into the category of representation learning on discrete representations of dynamic graphs. A main line of work in this category is based on RNNs. Typically, RNN-based methods achieve success, they have issues: 1) they suffer from same issues of RNNs that they cannot compress long-range dependencies into hidden states and they cannot be paralleled (Bahdanau and et al. 2015); 2) Methods like EvolveGCN (Pareja and et al. 2020) is similar to model-based meta-learning methods which use RNN to update the model parameters but recent research have found that such methods are more likely to overfit and have limited generalization ability compared to gradient-based meta-learning methods (Finn and Levine 2018).

In this paper, we argue that: a) the embeddings of dynamic graphs are formed by *time information* and *graph intrinsic information* (graph structure and attributes of nodes), where b) time information continuously changes with time and gives a prior on prediction targets, and graph intrinsic information directly contributes to the prediction targets.

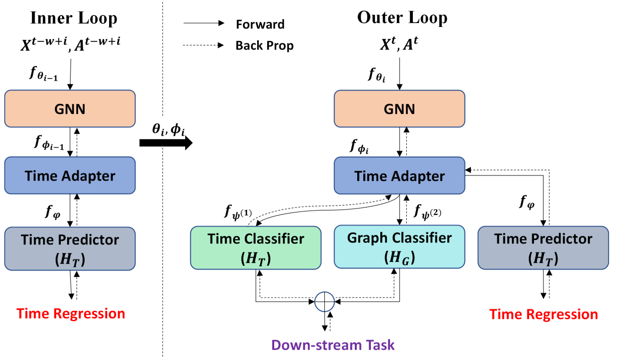

Figure 1: An illustration of our method. In inner loop, only GNN and time adapter are updated, while in outer loop, all the parameters are optimized.

We propose a novel algorithm Learning to Evolve on Dynamic Graphs (LEDG) as shown in Fig.1. First, we formulize our argument above by explicitly disentangling the embedding into *time embedding* and *graph intrinsic embedding*. The final prediction is performed by the combination of predictions on time embedding and graph intrinsic embedding. As the relative time between snapshots can be observed, we use a time predictor to predict the time by time embeddings to make sure that the embddings capture the time information. Second, we borrow the idea of gradient-based meta-learning (Finn and et al. 2017) and use episodic training to learn a model with the best initialization parameters that can quickly adapt to future graphs with only a small number of historical graphs. Our algorithm is model-agnostic and can be used for any message passing based GNN even if it is designed for static graphs in nature.

Our main contributions are as follows: (1) We propose a simple but effective attention-based method to disentangle the embeddings of dynamic graphs into time embeddings and graph intrinsic embeddings. (2) We propose a novel algorithm LEDG based on gradient-based meta-learning and can train any message passing based GNN on dynamic graphs. We perform detailed experiments and the results indicate that LEDG helps base model get higher performance.

*These authors contributed equally.
†Corresponding author.

Proposed Method

Problem Definition: In this paper, given a dynamic graph $\mathbb{G}$=$\{\mathcal{G}^1, \mathcal{G}^2, ..., \mathcal{G}^T\}$, representation learning on this graph aims to learn representations $\mathbf{H}^t$ of nodes at time t=$t_0, t_1, ..., T$, such that $\mathbf{H}^t$ can preserve both time information and graph intrinsic information, and thus can be used for down-stream tasks such as link prediction, edge classification, and node classification in future time.

Feature Disentanglement: We assume $\mathbf{H}$=$\mathbf{H}_G$+$\mathbf{H}_T$, where $\mathbf{H}_G$$\in$$\mathbb{R}^{N\times D}$ denotes the graph intrinsic embeddings and $\mathbf{H}_T$$\in$$\mathbb{R}^{N\times D}$ denotes the time embeddings. This assumption is reasonable as the original embeddings come from both the time and the graph. Given embeddings $\mathbf{H}$, we employ feature-wise attention to disentangle them into $\mathbf{H}_G$ and $\mathbf{H}_T$. A time adapter f_ϕ which is a multilayer perceptron (MLP) is used to get the attention map $\mathbf{S}$$\in$$\mathbb{R}^{N\times D}$ by $\mathbf{S}$=$\sigma\big(f_\phi(\mathbf{H})\big)$, where σ represents $Sigmoid$ function. The time embeddings $\mathbf{H}_T$ and graph intrinsic embeddings $\mathbf{H}_G$ are calculated by:

$$\mathbf{H}_G = \mathbf{S} \odot \mathbf{H}, and\, \mathbf{H}_T = (\mathbf{1} - \mathbf{S}) \odot \mathbf{H}, \tag{1}$$

where $\odot$ denotes Hadamard Product.

Time regression. To restrict $\mathbf{H}_T$ to best represent the temporal information, we use a time predictor denoted by f_φ which is an MLP to predict the current time t by $\mathbf{H}_T^t$ in this snapshot. We formulate the loss as:

$$\mathcal{L}_T\big(\mathbf{H}_T^t; f_\theta, f_\phi, f_\varphi\big) = smooth_{L_1}\big(f_\varphi(Pool(\mathbf{H}_T^t)) - t\big),$$
$$in\ which\ smooth_{L_1}(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \tag{2}$$

is a robust L_1 loss that is less sensitive to outliers than L_2 loss. $Pool$ denotes graph pooling.

Down-stream tasks. As the final prediction is related to both the time and the graph, we use two classifiers $f_{\psi^{(1)}}$ and $f_{\psi^{(2)}}$ which are two MLPs to calculate the predictions by:

$$\tilde{\mathbf{Y}} = Softmax(f_{\psi^{(1)}}(\mathbf{H}_T) + f_{\psi^{(2)}}(\mathbf{H}_G)). \tag{3}$$

Then the final loss can be calculated by cross-entropy loss:

$$\mathcal{L}_{CE}\left(\mathbf{H}_T, \mathbf{H}_G; f_\theta, f_\phi, f_\psi\right) = CrossEntropy\big(\tilde{\mathbf{Y}}, \mathbf{Y}\big), \tag{4}$$

where $\mathbf{Y}$ represents the ground truths of tasks.

Adapt feature extractor in inner loop. Specifically, for predicting snapshot at time t, we first initialize parameters $(\theta_0, \phi_0) \leftarrow (\theta, \phi)$ by the current model. Then we use SGD to update the parameters through w closest snapshots by Eq. (2). Denote index of snapshot in a time window as i=1, ..., w, the exact time of the snapshot is then t-w+i. Formally,

$$\begin{aligned}(\theta_i, \phi_i) \leftarrow\ & (\theta_{i-1}, \phi_{i-1}) \\ & - \eta_{in}\big[\nabla_{(\theta,\phi)}\big(\mathcal{L}_T\big(\mathbf{H}_T^{t-w+i}; f_{\theta_{i-1}}, f_{\phi_{i-1}}, f_\varphi\big)\big)\big],\end{aligned} \tag{5}$$

where η_{in} is the inner loop learning rate.

Update all parameters in outer loop. In each update step i of inner loop, we evaluate its performance on our target snapshot by Eq. (2) and Eq. (4), and formally,

$$\begin{aligned}(\theta,\ \phi, \psi, \varphi) \leftarrow & (\theta, \phi, \psi, \varphi) \\ & - \eta_{out}\nabla_{(\theta,\phi,\psi,\varphi)}\Big[\sum_{i=1}^{w}\Big(\mathcal{L}_{CE}\big(\mathbf{H}_T^t, \mathbf{H}_G^t; f_{\theta_i}, f_{\phi_i}, f_\psi\big) \\ & + \lambda\mathcal{L}_T\big(\mathbf{H}_T^t; f_{\theta_i}, f_{\phi_i}, f_\varphi\big)\Big)\Big],\end{aligned} \tag{6}$$

where η_{out} denotes the outer loop learning rate and λ is a hyperparameter that is used to balance the two losses.

Datasets	SBM		UCI		AS	
Metrics	MAP	MRR	MAP	MRR	MAP	MRR
GCN	.1894	.0136	.0001	.0468	.0019	.1814
GAT	.1751	.0128	.0001	.0468	.0200	.1390
GCN-GRU	.1898	.0119	.0114	.0985	.0713	.3388
EvolveGCN-H	.1947	$\underline{.0141}$	.0126	.0899	.1534	.3632
EvolveGCN-O	**.1989**	.0138	$\underline{.0270}$	.1379	.1139	.2746
DynGEM	.1680	.0139	.0209	.1055	.0529	.1028
dyngraph2vec V1	.0983	.0079	.0044	.0540	.0331	.0698
dyngraph2vec V2	.1593	.0120	.0205	.0713	.0711	.0493
LEDG-GCN	$\underline{.1960}$	**.0147**	**.0324**	**.1626**	$\underline{.1932}$	**.4694**
LEDG-GAT	.1822	.0123	.0261	$\underline{.1492}$	**.2329**	$\underline{.3835}$

Table 1: Link prediction results where mean average precision (MAP) and mean reciprocal rank (MRR) are displayed.

Experiment

Link Prediction: The results of link prediction are displayed in Table 1. Note that as our experiment setting in link prediction is the same as (Pareja and et al. 2020), for some of the baselines, we use the results reported in (Pareja and et al. 2020). Generally, GCN and GAT with our method significantly outperforms their vanilla versions in all datasets. Huge improvements are observed in datasets AS and UCI and LEDG with GCN performs better than all the baselines significantly, which proves the effectiveness of our method in improving the performances of base models on dynamic graphs. The MAP of SBM are similar for all the supervised methods while our method with GCN is with a bit higher MRR. We observe that in this task, our method with GCN outperforms that with GAT. We argue that the reason is that GAT is more likely to overfit under such setting as generalizing to future time requires high generalization ability.

Conclusion

We introduce a novel algorithm **LEDG** which is built on gradient-based meta-learning algorithm, for training GNNs on dynamic graphs. The algorithm learns updating strategies that have better generalization ability than RNNs. The core principle of our method is to disentangle the embeddings into time embeddings and graph intrinsic embeddings, and adapt the model parameters by time regression and down-stream tasks in a gradient-based meta-learning manner. The experiments demonstrate the effectiveness of our algorithm in training GNNs on dynamic graphs.

Acknowledgments

Supported by NSFC General Program (No.62176215).

References

Bahdanau, D.; and et al. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

Finn, C.; and et al. 2017. Model-agnostic Meta-learning for Fast Adaptation of Deep Networks. In *ICML*.

Finn, C.; and Levine, S. 2018. Deep Representations and Gradient Descent Can Approximate Any Learning Algorithm. In *ICLR*.

Pareja, A.; and et al. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *AAAI*.