Evolutionary Representation Learning for Dynamic Graphs

Aynaz Taheri [1] **Tanya Berger-Wolf** [1]

Abstract

Graph representation learning for static graphs is a well studied topic. Recently, a few studies have focused on learning temporal information in addition to the topology of a graph. Most of these studies have relied on learning to represent nodes and substructures in dynamic graphs. However, the representation learning problem for entire graphs in a dynamic context is yet to be addressed. In this paper, we propose an unsupervised representation learning architecture for dynamic graphs, designed to learn both the topological and temporal features of the graphs that evolve over time. The approach consists of a sequence-to-sequence encoder-decoder model embedded with gated graph neural networks (GGNNs). The GGNN is able to learn the topology of the graph at each time step, while LSTMs are leveraged to propagate the temporal information among the time steps. We demonstrate the efficacy of our approach by applying the obtained embeddings to dynamic graph classification using a real world dataset of animal behaviour.

1. Introduction

Dynamic graphs are a popular model for high level representation, characterization, and analysis of real world dynamic interaction systems. Dynamic graphs are often defined as time-ordered sequences of network snapshots (Holme & Saramäki, 2012), and each network snapshot models the interactions between the nodes over a unit interval of time. Graph representation learning approaches gained significant attention in downstream graph analysis applications in recent years. Several studies focused on learning to represent nodes and edges in dynamic graphs (Wang et al., 2017; Nguyen et al., 2018; Goyal et al., 2017). However, these approaches are still far from the ultimate goal of capturing both the topological and temporal features of entire time-varying graphs. Recently, remarkable achievements have been made in representation learning of whole static graphs (Taheri et al., 2018; Narayanan et al., 2016; Yanardag & Vishwanathan, 2015; Adhikari et al., 2017), but a comprehensive fundamental framework for processing dynamic graphs is still lacking. There are numerous differences between static and dynamic graphs that prevent direct transfer of methods for embedding static graphs to a dynamic setting. Many processes over dynamic graphs, such as information diffusion for example, depend on the temporal dynamics of these graphs, not just the topology of the connections. Therefore, an ideal approach should be able to learn both the topology and its temporal dynamics in order to represent a dynamic graph.

We propose an unsupervised learning approach for dynamic graph representation that combines the power of static graph representation methods and the success of recurrent neural networks in learning sequences of events through time. We investigate the extension of sequence-2-sequence encoder-decoder frameworks in learning to represent the temporal dynamics of a graph. We modify the encoder-decoder paradigm to propose a dynamic graph autoencoder by enforcing the decoder to reconstruct the dynamics of a graph, which have been observed by the encoder. We leverage the gated graph neural networks (GGNNs) (Li et al., 2016b) potential for learning the topology of the graph. GGNNs are embedded in a recurrent encoder to preserve the topology of a dynamic graph at each time step. In addition, a temporal information propagation module is incorporated to propagate the temporal information across nodes between consecutive time steps. Finally, an autoregressive decoder is leveraged to reconstruct the history of the graph evolution from the last encoder's hidden state.

We investigate whether the obtained representations from the dynamic graph autoencoder are generalized and sufficiently informative to be transferred to other downstream tasks, such as dynamic graph classification. To the best of our knowledge, this paper presents the first approach in learning to represent the entire graph in a dynamic setting. We demonstrate the efficacy of our approach in a dynamic classification task using the real-world dataset of animal behaviour. We provide several baselines to demonstrate the efficacy of our approach in the classification task and present quantitative analysis to indicate the impact of considering

[1]University of Illinois at Chicago, USA. Correspondence to: Aynaz Taheri <ataher2@uic.edu>.

Presented at the ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data

dynamics in our representation learning method.

2. Related Work

Recently, there has been a burst of methods for embedding nodes, subgraphs and even the entire graph (Narayanan et al., 2017; Taheri et al., 2018; Zhang et al., 2018; Lee et al., 2018) in a static setting into low-dimensional spaces. However, in this paper, we only focus on the related work for representation learning of graphs in a dynamic setting.

Lately, there have been a few studies on representation learning for dynamic graphs. The majority of these studies focused on representation learning for individual nodes within the dynamic graphs. Nguyen *et al.* (Nguyen et al., 2018) suggested using temporal random walks and the skip-gram model for learning node embeddings. Also, Du *et al.* (Du et al., 2018) proposed an extension of the skip-gram model in a dynamic setting to learn the representation of new nodes and adjust the old ones. Trivedi *et al.* (Trivedi et al., 2017) presented an architecture, Know-Evolve, for node representation learning in a temporal knowledge graph using recurrent neural networks and temporal point processes. DyRep (Trivedi et al., 2019) extended Know-Evolve with a two-time scale process that captures temporal node interactions in addition to the topological evolution. DyGEM (Goyal et al., 2017) incrementally computes node representations by initializing an autoencoder from the previous step. Goyal *et al.* (Goyal et al., 2018) extended DyGEM by adding recurrent neural autoencoders in order to capture more accurate temporal information of node embeddings. DynamicTriad (Zhou et al., 2018) imposed triad to learn the node embeddings while preserving the temporal information. Chen *et al.* (Chen et al., 2018) modified LSTMs in order to process dynamic networks and predict the upcoming links in the future. Wang *et al.* (Wang et al., 2018) introduced a small-scale method for representation learning of a series of transition graphs in the area of driving analysis. The approach is tuned for processing small-size graphs with a fixed number of nine nodes, and the entire adjacency matrix of the graph is flattened to a vector and used as the initial representation of the graph and input of the system. However, this approach does not scale to processing large-size, sparse, and variable number of nodes through time. In contrast, our approach focuses on the entire graph representation learning and preserves both topological and temporal properties of a graph.

3. Problem Definition

Dynamic graph: a dynamic graph is an ordered sequence of T graph snapshots: $G = \langle G_1, G_2, \ldots, G_T \rangle$. Graph $G_t = (V_t, E_t)$ models the state of a dynamic system at the interval $[t, t+\Delta t]$, for some fixed Δt. The dynamic graph G may have a subset of nodes from the set V at each time step, $V_t \subset V$. Each node $v \in V_t$ takes a unique identification value from $1, \ldots, |V|$, and edge $e^t_{kj} \in E_t$ is a pair of nodes $(k, j) \in \{E_t : V_t \times V_t\}$ and represents an edge at time step t.

G_t is represented by an adjacency matrix $A_t \in \mathbb{R}^{n \times n}$, where n is the number of nodes and where entry $a^t_{kj} = 1$ if there is an edge between nodes k and j at time step t, and $a^t_{kj} = 0$ otherwise.

Graph history: A sequence of graph snapshots which has been seen before time step t is called the history of G_t. We refer to the evolution history of a graph in the past w time steps by $H_{G_t} = \langle G_{t-w}, G_{t-w+1}, G_{t-w+2}, \ldots, G_{t-1} \rangle$.

Graph embedding: Given the H_{G_t}, we seek to learn a mapping function that embeds a graph G_t into $\mathbb{R}^d$ for some $d \in \mathbb{Z}$. The goal is to learn graph embeddings such that graphs with similar topological structure and temporal dynamics are close to one another in the embedding space. We focus on processing undirected graphs in this work, but it is trivial to extend our models to directed graphs.

4. Dynamic Graph Autoencoder

We investigate whether sequence-to-sequence encoder-decoder frameworks are capable of learning to represent a dynamic graph and capture the graph evolution through time. We leverage the GGNN's ability to capture the topology of a graph and couple it with the LSTM encoder-decoder architecture to capture the dynamics of the graph in order to create a dynamic network representation learning framework. We refer to this model as Dynamic Graph AutoEncoder, *DyGrAE*. The model is depicted in Figure 1.

GGNN: At first, the GGNN builds a graph representation for G_t by considering its topological structure at time step t. Let A denote the input adjacency matrix of a graph and we will use A_v to denote row v of that matrix. Let $x_v \in \mathbb{R}^k$ denote the initial embedding (node representation) of a node v, which is a draw from a uniform distribution. We use n^v_i to indicate the hidden state of a node v at iteration i. With this notation, we have the following propagation model. We initialize the hidden states of each node n^v_0 at the beginning as follows:

$$n^v_0 = x_v \tag{1}$$

Each propagation step passes information from the neighbors of a node v to learn its embedding a^v_i at propagation step i:

$$a^v_i = A_{v:}[n^1_{i-1} \ldots n^{|V|}_{i-1}] + b \tag{2}$$

where b is a bias. Note that after several iterations of this step the information is passed among all the reachable nodes to learn the embedding of each node.

The remaining equations are the update (Equation 3) and reset (Equation 4) gates of the GRU. The matrices W, U are

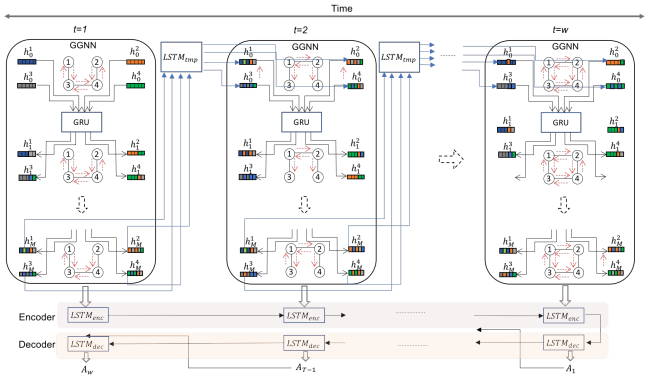

Figure 1. DyGrAE: Dynamic graph autoencoder

the parameter matrices for the GRU. The final embedding n_i^v of a node v at step i is given by Equation 6:

$$\text{Update: } z_i^v = \sigma(W^z a_i^v + U^z n_{i-1}^v) \quad (3)$$

$$\text{Reset: } r_i^v = \sigma(W^r a_i^v + U^r n_{i-1}^v) \quad (4)$$

$$\widetilde{n}_i^v = \tanh(W a_i^v + U(r_i^v \odot n_{i-1}^v)) \quad (5)$$

$$\text{Node state: } n_i^v = (1 - z_i^v) \odot n_{i-1}^v + z_i^v \odot \widetilde{n}_i^v \quad (6)$$

After M steps of message propagation in G_t, we use an average pooling of the nodes' hidden states (Equation 6) as the representation of the entire G_t:

$$Emb_{GGNN}(G_t) = Avg(n_M^v)$$

Temporal message propagation: We address the dynamic connectivity and reachability among nodes across different time steps in our model. Nicosia et al. (Nicosia et al., 2012) described the connectivity in dynamic graphs using the concepts of reachability and temporal (or time-respecting) paths. They also discussed that connectedness plays a significant role in temporal dynamics of a graph. A temporal path is a sequence of edges: $L = \langle e_{ij}^{l1}, e_{jk}^{l2}, \cdots, e_{mn}^{ln} \rangle$, with an increasing order of times $l1 \leq l2 \leq \cdots ln$ assigned to edges and where each node is visited exactly once. Node u is directly reachable from node v at time step t if there exists an edge uv at timestep t, *i.e.,* $a_{vu}^t = 1$. Node u is temporally reachable from node v if there is a temporal path from v to u. Temporal path generation cannot be addressed solely by the GGNNs since the message propagation process exists separately at each time step and there may be no connection among nodes across sequential time steps.

Our approach propagates messages not only in the topology of the graph at each time step, but also during a temporal window of graph evolution. To do so, we add a temporal message propagation module to the model using a recurrent neural network. $LSTM_{tmp}$ is incorporated to propagate the temporal information among the nodes through consecutive time steps. At each time step nodes' hidden states are initialized using the hidden states obtained by last iteration of the GGNN from previous time steps:

$$NodesInit(G_t) = LSTM_{tmp}(GGNN_{t-1}, h_{t-1}^{tmp})$$
$$GGNN_{t-1} = \{n_M^v(t-1) | v \in V\} \quad (7)$$

Here n_M^v is obtained by Equation 6. Information on a temporal path can be captured by $LSTM_{tmp}$, since $n_0^v(t)$ is initialized considering the $n_M^v(t-1)$, which is obtained by message propagation on the graph topology at time step $t-1$. Therefore, node v at time step t is able to receive messages from all other connected nodes through a temporal path.

Temporal encoder: Given the embedding of the graph $Emb_{GGNN}(G_t)$ and its history H_{G_t}, we use an LSTM encoder to project G_t into a hidden representation that takes the graph dynamics into account. The LSTM encoder, $LSTM_{enc}$, passes the information of the dynamic graph G over an observation window of size w and computes the dynamic embedding G_t using knowledge about the graph topology from the past w time steps:

$$h_t^{enc} = LSTM_{enc}(Emb_{GGNN}(G_t), h_{t-1}^{enc}) \quad (8)$$

We refer to h_t^{enc} as the dynamic embedding of the graph, $Emb_{dy}(G_t)$.

Temporal decoder: The main goal of the decoder, $LSTM_{dec}$, is to reconstruct the history of the graph H_{G_t} in the observed window with size w. The decoder is an autoregressive model and reconstructs the topology of the graph at time step t given the decoded graphs at previous time steps. The decoder uses h_w^{enc} to initialize its first hidden state. The set of edges of the graph is the output of the decoder at each time step.

$$h_t^{dec} = LSTM_{dec}(\overline{A}^{t-1}, h_{t-1}^{dec}) \quad (9)$$

We use a sigmoid transformation σ to reconstruct the adjacency matrix at time step $t-1$ (here W are the learned parameters and b is a bias):

$$\overline{A}^{t-1} = \sigma(h_{t-1}^{dec} * W + b) \quad (10)$$

Objective function: Let O be a set of temporal windows where each window $o \in O$ has a length of w. The parameter space of the models includes parameters related to $GGNN$, $LSTM_{enc}$, $LSTM_{dec}$, $LSTM_{tmp}$, $W \in \mathbb{R}^{|h^{dec}| \times |V|^2}$ and $b \in \mathbb{R}^{|V|^2}$. We use a cross entropy loss function to train our model:

$$Loss_{CE} = -\sum_{o \in O} \sum_{t=1}^{w} \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} A_{ij}^t \log(\overline{A}_{ij}^t)) + (1 - A_{ij}^t) \log(1 - \overline{A}_{ij}^t) \quad (11)$$

5. Experiments

In this section, we evaluate our dynamic representation learning approach using proposed models. We use the learned representation for a dynamic graph classification task. At each time step t, the graph G_t has a class label l from the set L. Our unsupervised learned representations for G_t are used in the classification task.

Baselines

We compare our results with several baselines. Li *et al.* (Li et al., 2016a) proposed an Adversarial Sequence Tagging (AST) to perform activity labeling classification. Amornbunchornvej *et al.* (Amornbunchornvej et al., 2018) used dynamic graph features at each time step to predict the activity label using linear discriminant analysis (LDA). We also compare with two other baselines that learn static graph representations. Taheri *et al.* (Taheri et al., 2018) proposed an unsupervised sequence-to-sequence (S2S) approach for static graph representation learning. Moreover, we use a supervised GGNN classifier as a baseline to classify G_t. We also compare our models with some of the temporal node embedding approaches, mentioned in Section 2: *DynGEM*(Goyal et al., 2017), *DyAE*, *DyAE*, *DyRNN* and *DyAERNN* from *dynamic2vec* framework(Goyal et al., 2018) .

Dataset

We used the dataset of dynamic graphs of a troop of GPS-tracked baboons living in the wild in Mpala Research Centre, Kenya (Strandburg-Peshkin et al., 2015; Crofoot et al., 2015). Due to the availability of the behavior labels provided by biologists, we used a subset of 2 days (out of 28 days) of 16 adult and sub-adult members of the troop that were fitted with GPS collars, collecting data points at 1Hz for 12 hours (6am to 6pm). Amornbunchornvej *et al.* (Amornbunchornvej et al., 2018) constructed dynamic graphs of the individuals and performed the activity classification task. The dynamic graph of the first day has 23259 time steps and the second day has 19098 time steps. There are four group level activities in the baboon dataset: sleeping, hanging-out, coordinated non-progression, and coordinated progression. One of the four labels is assigned to the dynamic graph at each time step by the domain experts. We report two results, following the configuration baselines in (Amornbunchornvej et al., 2018; Li et al., 2016a): using the labeled first day of data to classify the second days activities (Baboon (day 1)); and using the second days labeled data to classify the first days activities (Baboon (day 2)).

Hyperparameters

The GGNN uses four iterations of message passing at each time step. The dimensionality of the GRU and LSTMs is fixed to 100. The Adam optimizer is used for minibatch training. We set the learning rate to be 0.001 and dropout rate to 0.5. Figure 2 indicates the effect of window size and batch size on the classification performance. It shows that the best accuracy is obtained by larger window and batch sizes, which means that longer-term dynamics matter for this task. Having larger window sizes causes an explicit larger context for representation learning during backpropagation, and having larger batch sizes causes an implicit larger context through time. Figure 2 shows that we obtain the best results with window sizes larger than 50 and the batch size larger than 50, which we used in the rest of our experiments.

Method	Baboon Day 1	Baboon Day 2
AST	77.30	69.22
LDA	87.20	70.82
GGNN	84.58	81.38
S2S	84.66	83.87
DyGEM	57.09	47.36
DyAE	80.28	72.21
DyRNN	63.21	52.9
DyAERNN	62.73	52.0
DyGrAE	88.83	87.54

Table 1. Classification accuracy of the first two days of baboon data

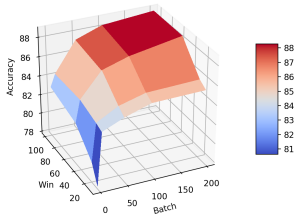

Figure 2. The effect of window size and batch size on accuracy

Results

We used a C-SVM classifier from LIBSVM with a radial basis kernel for multi-class classification. The data of one day is used for training and tuning the regularization and kernel hyperparameters of the SVM via cross-validation and the other day is used for testing. We compare our proposed models with other baselines in Table 1. We exceed all other baselines and the comparison with static graph approaches (*GGNN* and *S2S*) shows that dynamics matter and incorporating temporal information in embedding methods improves graph representations. Moreover, the results demonstrate that temporal node embedding approaches (*DyGEM*, *DyAE*, *DyRNN* and *DyAERNN*) do not necessarily contribute to informative representations of the whole graph at each time step. We speculate that the inefficiency of these approaches originates from the fact that these autoencoders are trained to reconstruct a node and its incident edges (the node's corresponding row in adjacency matrix), not the whole topology of the graph.

6. Conclusions

We proposed an efficient approach for representation learning of entire graphs in a dynamic setting. We evaluated our approach on a real-world dataset of animal behaviour and show that our model achieves significantly better performance compared to the state-of-the-art models that learn the graph representation in a static setting. The promising results indicate that our approach can be used as the basis for dynamic graph analysis.

References

Adhikari, B., Zhang, Y., Ramakrishnan, N., and Prakash, B. A. Distributed representations of subgraphs. In *DaMNet*, 2017.

Amornbunchornvej, C., Brugere, I., Strandburg-Peshkin, A., Farine, D. R., Crofoot, M. C., and Berger-Wolf, T. Y. Coordination event detection and initiator identification in time series data. *ACM Trans. Knowl. Discov. Data*, 12 (5):53:1–53:33, 2018.

Chen, J., Xu, X., Wu, Y., and Zheng, H. Gc-lstm: Graph convolution embedded lstm for dynamic link prediction. *arXiv*, 2018.

Crofoot, M., Kays, R., and M, W. Data from: Shared decision-making drives collective movement in wild baboons. *Movebank Data Repository*, 2015.

Du, L., Wang, Y., Song, G., Lu, Z., and Wang, J. Dynamic network embedding: An extended approach for skip-gram based network embedding. In *IJCAI*, 2018.

Goyal, P., Kamra, N., He, X., and Liu, Y. Dyngem: Deep embedding method for dynamic graphs. *CoRR*, abs/1805.11273, 2017.

Goyal, P., Chhetri, S. R., and Canedo, A. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *arXiv*, 2018.

Holme, P. and Saramäki, J. Temporal networks. *Physics reports*, 519(3), 2012.

Lee, J. B., Rossi, R., and Kong, X. Graph classification using structural attention. In *KDD*, pp. 1666–1674. ACM, 2018.

Li, J., Asif, K., Wang, H., Ziebart, B. D., and Berger-Wolf, T. Y. Adversarial sequence tagging. In *IJCAI*, 2016a.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. Gated graph sequence neural networks. In *ICLR*, 2016b.

Narayanan, A., Chandramohan, M., Chen, L., Liu, Y., and Saminathan, S. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. *MLG*, 2016.

Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S. graph2vec: Learning distributed representations of graphs. In *MLG*, 2017.

Nguyen, G. H., Lee, J. B., Rossi, R. A., Ahmed, N. K., Koh, E., and Kim, S. Continuous-time dynamic network embeddings. In *International Workshop on Learning Representations for Big Networks at The Web Conference*, 2018.

Nicosia, V., Tang, J., Musolesi, M., Russo, G., Mascolo, C., and Latora, V. Components in time-varying graphs. *Chaos: An interdisciplinary journal of nonlinear science*, 22(2), 2012.

Strandburg-Peshkin, A., Farine, D. R., Couzin, I. D., and Crofoot, M. C. Shared decision-making drives collective movement in wild baboons. *Science*, 348(6241):1358–1361, 2015.

Taheri, A., Gimpel, K., and Berger-Wolf, T. Learning graph representations with recurrent neural network autoencoders. In *KDD Deep Learning Day*, 2018.

Trivedi, R., Dai, H., Wang, Y., and Song, L. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *ICML*, 2017.

Trivedi, R., Farajtabar, M., Biswal, P., and Zha, H. Dyrep: Learning representations over dynamic graphs. In *ICLR*, 2019.

Wang, P., Fu, Y., Zhang, J., Wang, P., Zheng, Y., and Aggarwal, C. You are how you drive: Peer and temporal-aware representation learning for driving behavior analysis. In *KDD*. ACM, 2018.

Wang, Q., Mao, Z., Wang, B., and Guo, L. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12), 2017.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *KDD*, 2015.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018.

Zhou, L.-k., Yang, Y., Ren, X., Wu, F., and Zhuang, Y. Dynamic network embedding by modeling triadic closure process. In *AAAI*, 2018.