Computing and Informatics, Vol. 41, 2022, 1491–1509, doi: 10.31577/cai_2022_6_1491

DYNAMIC NETWORK REPRESENTATION LEARNING METHOD BASED ON IMPROVED GRU NETWORK

Jianguo Pan, Huan Li, Jiajun Teng, Qin Zhao*

College of Information, Mechanical and Electrical Engineering
Shanghai Normal University
Shanghai 201418, China
e-mail: {panjg, q_zhao}@shnu.edu.cn, {hlhl258000, ccj258000}@163.com

Maozhen Li*

College of Information, Mechanical and Electrical Engineering
Shanghai Normal University
Shanghai 201418, China
&
Department of Electronic and Computer Engineering
Brunel University London
Uxbridge, UB8 3PH, UK
e-mail: maozhen.li@brunel.ac.uk

Abstract. As social networks have been rapidly growing, traditional network representation learning methods are struggling to accurately characterize their dynamic changes, and to output effective node classification and link prediction. To address this problem, this paper proposes IproGRU, a dynamic network representation learning method based on an improved Gated Recurrent Unit (GRU) network to improve the dynamic network representation. First, the method quickly generates embedding for an influenced node by sampling and aggregating features of its neighboring nodes when the network changes. Second, it updates the embedding of the influenced node on time series by the improved GRU network to fully adapt to the changes of the dynamic network. Experimental results on node classification and link prediction for three datasets of dynamic networks show that the proposed method improves the accuracy by 5–10 % on average from those of the traditional

* Corresponding authors

Node2vec and GraphSAGE methods and has a slight advantage over Graph Convolutional Networks (GCNs). The results demonstrate that our method is effective for dynamic network representation.

Keywords: Dynamic networks, GRU, node classification, link prediction

1 INTRODUCTION

The "social network", a sociological concept proposed by J. A. Barnes, is used to describe human relations [1]. A social network is a structure composed of nodes and edges, where the nodes represent individuals or organizations and the edges represent certain behaviors or relationships between the nodes. A social network can be modeled and analyzed by graphs for studies such as node classification and link prediction. With the rapid development of social networks, their features, including the number and attributes of nodes, the related information of links, etc., are dynamically changing over time. And these changes have led to the rapid expansion of data scales. The changing features and the increasing data scales have imposed more challenges on accurate representation and analysis for social networks. Therefore, how to accurately represent dynamic networks has become a key issue for studies on social networks.

Traditional representation methods can neither fully capture the dynamic changes of network structures, nor accurately represent current states of those structures. In a social network, adding or deleting a node or a link can cause changes of the whole network, and constant changes of nodes and links will deviate the network from its original structure. But constantly and repeatedly learning the whole changing network will consume exceptionally much more resources and cannot obtain an accurate representation of the social network.

To cope with such dynamic changes in social networks, a dynamic network representation based on an improved GRU network (IproGRU) is proposed in this paper. Inspired by Graph Neural Networks (GNNs) [2, 3, 4, 5, 6, 7], a vector representation of a node is obtained by aggregating feature information of their neighboring nodes, and the improved GRU network is used to extract the spatial features in the dynamic network. In this way, the dynamic learning process of the social network over time is established to obtain an accurate network representation.

2 RELATED WORK

On the social platforms like Sina Weibo, users and their friends form typical social networks. These networks contain important information that can facilitate functions such as friend recommendation, community discovery, and user modeling. Network representation learning, also known as network embedding, is an important way to mine social networks for information. Its core is to embed unstructured data, such as nodes, links or communities, into a low-dimensional space, and represent nodes, links or even the whole network with low-dimensional vectors [8]. Thus, the unstructured data are transformed into structured ones, while the original information is preserved as much as possible to support subsequent applications such as node classification [9], link prediction [10, 11], and community discovery [12].

Traditional representation learning methods include the Principal Component Analysis (PCA) [13], the Multidimensional Scaling (MDS) [14], the Isometric Mapping (Isomap) [15] and the Local Linear Embedding (LLE) [16]. These methods preserve the overall structure of nonlinear manifolds with a high time complexity and are only suitable for small networks. The network representation learning methods proposed in recent years are built on previous node embedding methods and recent developments in GNNs for graph-structured data. They intend to preserve certain features of the original graph in the embedding space. DeepWalk [17] is the first method that integrates deep learning into network embedding. By treating nodes as words and generating short random walks as sentences, one can bridge the gap between networks and word embeddings. Neural network models, such as Skip-gram [18], can then be applied to these random walks to obtain network embeddings. Node2vec [19] improves DeepWalk's random walk strategy by using bias parameters to decide the choice between the Depth-First-Search or the Breadth-First-Search. Large-scale Information Network Embedding (LINE) [20] compensates for the sparsity of the DeepWalk's first-order nearest-neighbors by using the rich second-order nearest-neighbor relations. GraRep [21] is an alternative approach to random walks via matrix decomposition and provides a good representation for weighted networks. However, as networks expand, GraRep will cost increasingly computational resources and its representation performance is unlikely to be ideal.

These traditional representation methods focus on the representation of static networks and cannot adapt to the changes in the network. In order to capture the new information that the changes bring, these methods must be retrained with significant extra time and resource consumption. Thus, some studies have attempted to learn node representations in dynamic graphs by applying a temporal regularizer to enforce smoothness of the node representation from adjacent snapshots [22, 23]. But these methods can hardly deal the situation where the nodes exhibit significantly distinct evolutionary behaviors. GraphSAGE [24] and FastGCN [25] were developed for fast representation learning on graphs. Specifically, GraphSAGE computes node representations by sampling the neighborhood of each node and then executes a specific aggregator for information fusion. The FastGCN method interprets the convolution of the graph as an integral transformation of the embedding function and samples the nodes at each layer independently. Continuous-Time Dynamic Network Embeddings (CTDNE) [26] is based on DeepWalk and builds more sensible sampling paths by modifying the path sampling method so that the paths meet the actual order of occurrence. Dynamic Graph Transformer (DGT) [27] uses spatial-temporal encoding to effectively learn graph topology and capture implicit links. Graphormer [28] uses scaled dot-product attention [29] for message aggregation and extends the idea of positional coding to the graph domain. But it is only feasible to small molecule graphs and cannot cope with large graphs due to the significant computation cost of full attention.

3 THE PROPOSED METHOD

Targeting the dynamic changes in social networks, this paper proposes IproGRU. This method is developed based on the Barabási-Albert [30] network model since nodes and links in dynamic networks change continuously with time. The working process of the method consists of three procedures as follows:

1. The dynamic representation of an influenced node is generated by sampling and aggregating the features of its neighboring nodes. The method samples the k layers neighboring nodes of the influenced node, merges the features of the neighboring nodes through an aggregation function, and then merges the features of the influenced node with the features of the aggregated neighboring nodes for the purpose of learning the features of the sampled node.
2. This method updates the representation with a series of dynamic changes in the network structure. Unlike other dynamic network representation learning methods, the neural network designed in this paper is based on an improved GRU. It can update the node sequence changes with temporal information, and the dynamic change process of the whole network can be recorded and described.
3. It has been shown that IproGRU is optimized by increasing the similarity of neighboring node representations, since closer neighboring nodes should have more similar embedding representations.

   The method's workflow is shown in Figure 1.

3.1 Problem Definition

Given a graph structure $G = (V, E)$ of a social network, V and E are the vertex set and the edge set, respectively. The dynamic change process of the network can be described as a series of interactive information under time t and symbolized as (v_i, v_j, t). Our goal is to update the representation of the whole network with this series of temporal information. In other words, after the topology information of the dynamic network is captured, the attributes of each node are mapped to a continuous low-dimensional vector space R^d. Thus, the representation of the whole dynamic network can be obtained effectively for the subsequent application tasks.

In the real world, the social networks never remain static and continue to change with time. Therefore, before building the dynamic network representation method, it is necessary to identify possible changes of the real-world social networks:

1. Emergence of a new independent node: a new user joins a social network, but does not link to any existing user in the network.

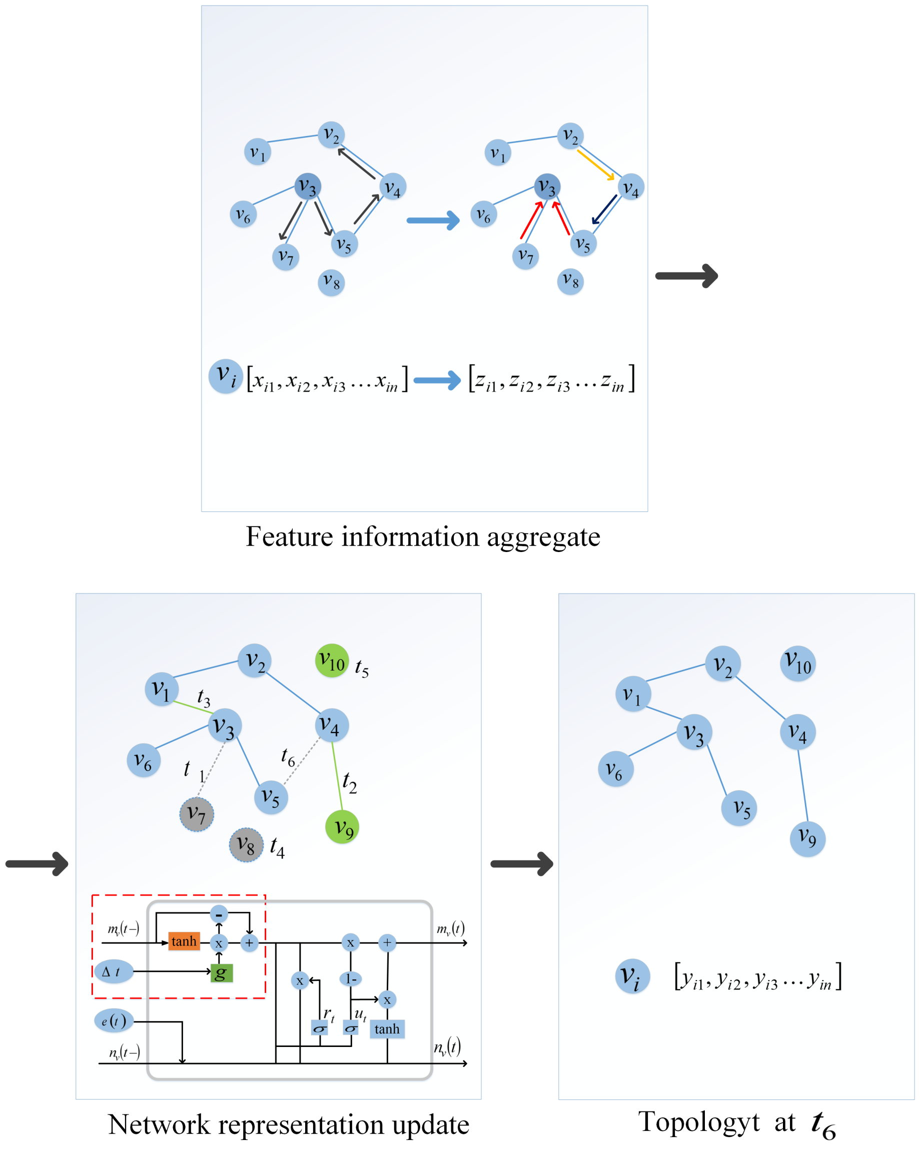
$v_i\ [x_{i1}, x_{i2}, x_{i3} \ldots x_{in}] \rightarrow [z_{i1}, z_{i2}, z_{i3} \ldots z_{in}]$
Feature information aggregate
$m_v(t-)$
Δt
$e(t)$
$n_v(t-)$
$m_v(t)$
$n_v(t)$
r_t
u_t
tanh
Network representation update
$v_i\ [y_{i1}, y_{i2}, y_{i3} \ldots y_{in}]$
Topologyt at t_6

Figure 1. The workflow of IproGRU

2. Emergence of a new link: a new link can be created between two existing users who were not directly related before and are now connected because of common friends, interests, hobbies, or some events.
3. Emergence of a new dependency node: a new user joins the social network and simultaneously establishes a link with one or more existing users.
4. Disappearance of an independent node: an isolated user exits the social network.
5. Disappearance of a link: an existing link between two users is terminated.
6. Disappearance of a node: a user who has links to other users quits the social network and all their links are terminated.

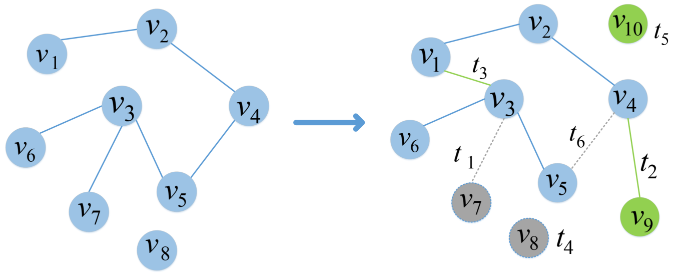

Figure 2. An example of the dynamic changes in a social network graph

Figure 2 illustrates the dynamic changes in a social network. Emergence and disappearance of an independent node does not affect the representation of other nodes, and only the embedding of the node needs to be added to or removed from the network. The other four cases involve creations and removals of links and nodes those links connect. We regard these nodes as the influenced nodes in this paper and IproGRU is designated only to the influenced nodes.

Therefore, we consider the four changes with link changes in order to better capture the representation of the network and describe its dynamic changes over time. The IproGRU method consists of two parts: feature information aggregation and network representation update. And an optimization method is developed to finesse the similar representations between nodes that are closer together.

3.2 Feature Information Aggregate

The first step in the dynamic network representation method is to sample and aggregate neighborhood features for the influenced nodes. The nodes considered in this paper are not isolated, they are more or less connected to other nodes in the network. When a new dependency emerges, i.e., an unlabeled node joins the network and is connected to one or more other nodes at the same time, then, how to quickly generate embedding for that node is an important question. This paper proposes a new method that aggregates the feature information of the influenced node's

k layers neighbors when the network changes, and merges the aggregated information into the features of that influenced node, so that when a new node is added or other changes occur on the node, the influenced node can learn the information of the whole network quickly through its neighbors.

Algorithm 1 defines the process of the feature information aggregation. Motivated by [24], the parameters are fixed. We assume that K aggregation functions are learned for each node, and denoted as $AggF_k$, $\forall k \in \{1, 2, \ldots, K\}$. They gather the information of the k layers neighboring nodes, and W_k, $\forall k \in \{1, 2, \ldots, K\}$ is a set of weight matrices for propagating information among neighboring nodes of different layers.

Algorithm 1 Feature information aggregation algorithm

```
Input: Graph $G = (V, E)$; input features $\{x_v, \forall v \in V\}$; depth $K$;
weight matrices $W_k$, $\forall k \in \{1, 2, \ldots, K\}$; non-linearity $\sigma$;
aggregator function $AggF_k$, $\forall k \in \{1, 2, \ldots, K\}$;
neighborhood sampling collection $N_k, \forall k \in \{1, 2, \ldots, K\}$
Output: vector representations $z_v$ for all $\forall v \in V$
1  $h_v^0 \leftarrow x_v, \forall v \in V$;
2  for $k = 1, \ldots, K$ do
3     for $\forall v \in V$ do
4        $h_{N(v)}^k \leftarrow AggF_k^{pool}(\{h_u^{k-1}, \forall u \in N_k(v)\})$;
5        $h_v^k \leftarrow \sigma(W_k \cdot concat(h_v^{k-1}, h_{N(v)}^k)$;
6     end
7     $h_v^k \leftarrow h_v^k / ||h_v^k||_2, \forall v \in V$;
8  end
9  $z_v \leftarrow h_v^K, \forall v \in V$;
```

In Algorithm 1, after each iteration, the representation of the current node includes both its own attributes and the feature information of its neighboring nodes in the corresponding layer. This aggregation algorithm therefore enriches the representation of the node.

The input of Algorithm 1 consists of $G = (V, E)$, and the feature information of the nodes in network $x_v, \forall v \in V$. A sampling algorithm can be used to sample and extract the neighboring nodes in 1 to k layers. Then, the feature information of the sampled neighboring nodes is aggregated (the fourth line of the Algorithm 1). Motivated by [24], a pool aggregation structure is built, $AggF_k^{pool} = \max(\{\sigma(W_{pool} h_{u_i}^k + b), \forall u_i \in N(v)\})$, where max denotes the element-wise max operator and σ is nonlinear activation function. Then, the node's current representation, h_v^{k-1}, and the aggregated neighborhood vector, $h_{N(v)}^k$, are fed to a fully connected layer with the nonlinear activation function σ, which transforms representations of the node (the fifth line of Algorithm 1). Finally, the vectorization representation of node v is output. It is worth noticing that the neighboring nodes in different layers influence the information propagation of the influenced node v differently. When aggregating the feature information of neighboring nodes into the influenced node, the neighboring nodes that are closer to the influenced node are assigned with larger weights.

3.3 Network Representation Update

After merging the feature information of neighboring nodes, the influenced nodes should be updated on the time series using the update component. The order of node changes is important for the formation of node representations. For example, in e-commerce, the most recent purchase record is more likely to capture the latest user preferences than the older records. In this paper, we view each node change over time as a 'sequence'. Instead of recording all the information in this 'sequence', we only keep the most recent information about the node, and update the sequence when the node is influenced. A series of changes in a dynamic network may involve different changes to the same node at multiple times, and the output of the update operation of the previous change to the influenced node should be an input to its next change. For example, if node v_3 is disconnected from node v_7 at t_1 and is linked to node v_1 at t_3, the output of node v_3 at time t_1 is one of its inputs at time t_3.

Changes in the network can be represented by $(v_i, v_j, t, \mathit{flag})$, where v_i, v_j are the influenced nodes, t refers to the time when the change occurs and *flag* indicates whether the change leads to an expansion or shrinking of the network. $\mathit{flag} = 1$ represents the expansion of the network and indicate the second and the third changes listed in Section 3.1. In Figure 2, a set of changes $(v_1, v_3, t_3, 1)$ occurs at t_3 and the influenced nodes are v_1 and v_3. $\mathit{flag} = -1$ represents the shrinking of the network and denotes the last two changes listed in Section 3.1. For example, in Figure 2, at t_6 the link between v_4 and v_5 disappears and the influenced nodes are v_4 and v_5, so $(v_4, v_5, t_6, -1)$ can be used to describe this change at t_6. Since the time series is a critical parameter to the sequence of update operations, the proposed method is based on an improved GRU to perform update operations. A GRU is a variant of Long Short-Term Memory (LSTM) [31].

Algorithm 2 Network representation update algorithm

```
Input: $\mathit{flag}$; vector representations $z_{v_i}(t)$ and $z_{v_j}(t)$; cell memory $m_v(t-)$;
short-term cell memory $m_v^s(t-)$; long-term cell memory $m_v^l(t-)$; hidden
state $n_v(t-)$; time interval $\Delta t$; decreasing function $g$; weight matrices $W_1$
and $W_2$; activation function $act(\cdot)$ etc.
Output: cell memory $m_v(t)$; hidden state $n_v(t)$
1   $e(t) \leftarrow act(W_1 \cdot z_{v_i}(t) + W_2 \cdot z_{v_j}(t) + \varepsilon)$;
2   use $g(\Delta t)$ to reduce $m_v^s(t-)$;
3   $m_v^*(t-) \leftarrow m_v^s(t-), m_v^l(t-)$;
4   $m_v(t), n_v(t) \leftarrow \mathit{standard\ GRU}(e(t), m_v^*(t-), n_v(t-), \mathit{flag})$;
```

Algorithm 2 defines the process of updating the network representation. It uses a feedforward neural network to model the information of the directly influenced nodes before the update operation:

$$e(t) = act(W_1 \cdot z_{v_i}(t) + W_2 \cdot z_{v_j}(t) + \varepsilon) \tag{1}$$

where $z_{v_i}(t)$ and $z_{v_j}(t)$ are representations of nodes v_i and v_j, respectively, after they have merged the features of neighboring nodes of k layers. W_1, W_2 and ε are parameters of the neural network and $act(\cdot)$ is an activation function such as *sigmoid* or tanh. Output $e(t)$ contains the sequence information of the nodes, $(v_i, v_j, t, flag)$.

The update operation of a node depends heavily on recent impacts, while earlier impacts are forgotten to varying degrees along the timeline. Therefore, in order to better establish update operations, the GRU is modified to the structure shown in Figure 3, based on [32, 33].

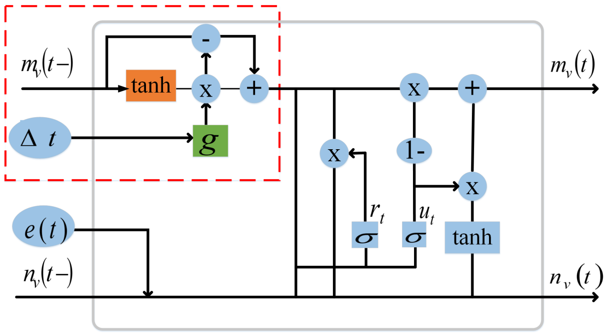

Figure 3. The improved GRU for network representation update

The input to the update method is the most resent cell memory $m_v(t-)$, the hidden state $n_v(t-)$, the time interval Δt and the information about the influenced node $e(t-)$. The output is the cell memory $m_v(t)$ and the hidden state $n_v(t)$ of the influenced node.

The difference between the improved GRU and the standard GRU is indicated in the dashed box in Figure 3. These different parts of the operation are described as:

$$m_v^s(t-) = \tanh(W_d \cdot m_v(t-) + \lambda_d), \tag{2}$$

$$\widehat{m_v^s}(t-) = m_v^s(t-) * g(\Delta t), \tag{3}$$

$$m_v^l(t-) = m_v(t-) - m_v^s(t-), \tag{4}$$

$$m_v^*(t-) = m_v^l(t-) + \widehat{m_v^s}(t-). \tag{5}$$

The node's cell memory $m_v(t-)$ is divided into a short-term cell memory $m_v^s(t-)$ and a long-term cell memory $m_v^l(t-)$. The former represents the discounting (forgetting) of node information over time, while the latter is generated by the neural network. In Equation (3), Δt denotes the time interval between two adjacent times when the node is influenced, $\widehat{m_v^s}(t-)$ denotes that the short-term cell memory is forgotten after a time interval Δt, and g is a decreasing function. In Equation (5), $m_v^*(t-)$ denotes the combination of the long-term cell memory $m_v^l(t-)$ and the discounted short-term cell memory $m_v^s(t-)$. It is the adjusted cell memory and can be considered as the output of the dashed box that is fed into the standard GRU cell (the rest of the update cell).

The other operations in Figure 3, same as those in the standard GRU [34], are defined as follows:

$$r_t = \sigma(\mathit{flag} \cdot W_r \cdot e(t) + W_r \cdot n_v(t-) + \lambda_r), \tag{6}$$

$$u_t = \sigma(\mathit{flag} \cdot W_z \cdot e(t) + W_z \cdot n_v(t-) + \lambda_z), \tag{7}$$

$$\widetilde{m_v}(t) = \tanh(\mathit{flag} \cdot W_{\widetilde{m}} \cdot e(t) + W_{\widetilde{m}} \cdot (r_t * n_v(t-)) + \lambda_{\widetilde{m}}), \tag{8}$$

$$m_v(t) = (1-u_t) * m_v^*(t-) + u_t * \widetilde{m_v}(t), \tag{9}$$

$$n_v(t) = \sigma(W_0 \cdot m_v(t)). \tag{10}$$

For convenience, the procedure for updating the modules is summarized as follows:

$$m_v(t), n_v(t) = \mathit{Update}(e(t), m_v(t-), n_v(t-), \Delta t, \mathit{flag}). \tag{11}$$

3.4 Optimization Method

By aggregating the information of neighboring nodes and updating the representation of the network, we enrich the embedding representation of the nodes in the network. The nodes in the network can better reflect the spatial and the temporal features of the whole network. This kind of representation is closer to the reality and can help the network more effectively adapt to the dynamic changes. For example, if a new node in a social network has no label information, a node embedding representation can be quickly generated through its neighboring nodes' feature information. It has been shown [19, 24] that in real networks, the closer the connections between nodes, the more similar the properties of the nodes.

In this regard, we propose an optimization method to obtain a more similar embedding representation of nodes that are close to each other. This method uses the parameters involved in aggregating the neighbor information (Algorithm 1) by means of the loss function:

$$J(z_v) = -\log(\sigma(z_v^T z_u)) - Q \cdot E_{u_n \sim P_n(u)} \log(\sigma(-z_v^T z_{u_n})), u \in N_v \tag{12}$$

where z_v is the embedding representation of the influenced node; Q defines the number of negative samples; p_n is a negative sampling distribution; v is the influenced node; and N_v is the set of sampled neighbors of node v. The loss function encourages nearby nodes to have similar representations.

4 EXPERIMENTAL SETUP

4.1 Datasets

We chose three datasets to evaluate IproGRU. Two of them were gleaned from real online social platforms: HEP-TH and Epionios, and the other one is a graph network repository: BlogCatalog. The features of the three datasets are listed in Table 1.

	HEP-TH	Epinions	BlogCatalog
# Node	1 424–7 980	14 180	10 312
# Edges	2 556–21 036	227 642	333 983
# Time Steps	60	100	60

Table 1. Statistical information of experimental data sets

HEP-TH [35]: HEP-TH contains abstracts of academic papers in Conference on Theory of High Energy Physics from January 1993 to April 2003. We prepared the data in the same way as [37] did: for each month, we created a collaborative network using all papers published up to that month; and we built a time series of 60 graphs from the first five years data where the nodes were increased from 1 424 to 7 980.

Epinions [36]: Epinions is a product review site in which users share their reviews and opinions about products. Users themselves can also build trust networks to seek advice from others. The network contains 14 180 nodes and 227 642 edges, Epinions is very useful for experiments, especially for those on the combination of recommendation systems and social interactions.

BlogCatalog [37]: BlogCatalog is a social blog directory in which users can post their blogs under different predefined categories. This dataset includes 10 312 nodes and 333 983 edges, and the label dimension is 39 [19]. BlogCatalog is a static property dataset, in order to implement a dynamic environment, we randomly added 0.1 % of new edges and changed 0.1 % of the attribute values at each time step to simulate the evolution process [36].

4.2 Baselines

We deployed three classic social graph analysis methods for the tasks of node classification and link prediction, thus the results could be adequately compared. The three methods are illustrated as follows:

GraphSAGE is an inductive learning method that uses the vertex attribute information to efficiently generate unknown vertex embedding. The core idea is to generate the target node embedding vector by learning a function that aggregates neighbor nodes.

Node2vec is a graph embedding method [19]. First, it incorporates Depth-First-Search neighborhoods and Breadth-First-Search neighborhoods that directly imports the word2vec package sample through a specific walking method, and generates a corresponding sequence for each point. Second, it treats these sequences as texts and imports them into a Skip-gram method in word2vec to obtain the vector of each node (corresponding to the vector of each word in word2vec).

GCNs [39] is a state-of-the-art graph convolutional network method. It attempts to learn better node features by aggregating information from the node's neighbors. Since the method cannot use temporal information, it regards the dynamic graphs as static ones in our experiments by ignoring the time of edge creation.

4.3 Evaluation Metrics

In the node classification task, TP(A), FP(A), TN(A) and FN(A) were used to indicate the number of true positives, false positives, true negatives and false negatives, respectively, among the instances predicted to be label A. C is the overall set of labels.

Accuracy (AC) was used to measure the proportion of the entire sample space that was correctly classified, defined as follows:

$$AC = \frac{\sum_{A\in C}(TP(A) + TN(A))}{\sum_{A\in C}(TP(A) + TN(A) + FP(A) + FN(A))}. \tag{13}$$

Macro-F1 is a metric that assigns the same weight to each class [40]. It is defined as follows:

$$\text{Macro-F1} = \frac{\sum_{A\in C} F1(A)}{C} \tag{14}$$

where F1(A) is the F1-measure for the label A.

Micro-F1 is a metric that assigns the same weight to each instance. It is defined as follows:

$$\text{Micro-F1} = \frac{2 * P_r * R}{P_r + R} \tag{15}$$

where $P_r = \frac{\sum_{A\in C} TP(A)}{\sum_{A\in C}(TP(A)+FP(A))}$ and $R = \frac{\sum_{A\in C} TP(A)}{\sum_{A\in C}(TP(A)+FN(A))}$.

In the link prediction task, Mean Average Precision (MAP) was used as an evaluation metric for link prediction. MAP is a metric with good discrimination and stability, according to reference [41], it can be written as:

$$\text{MAP} = \frac{\sum_i AP(i)}{|V|} \tag{16}$$

where $AP(i) = \frac{\sum_k precision@k(i) \cdot \prod\{E_{pred_i(k)} \in E_{gt_i}\}}{|\{k : E_{pred_i}(k)\} \in E_{gt_i}|}$ and $P@k = \frac{|E_{pred}(k) \cap E_{gt}|}{k}$. $P@k$ is the fraction of correct predictions in the top k predictions. E_{pred} and E_{gt} are the predicted and ground truth edges, respectively. MAP averages the precision of all nodes. High MAP values imply that the method can achieve good predictions.

5 EXPERIMENTS

We conducted a series of experiments to evaluate the effectiveness of IproGRU. Specifically, node classification and link prediction tasks were designed on three different datasets to verify the practicality of IproGRU. A comparison between IproGRU and certain classical embedding methods was also conducted.

5.1 Node Classification

We conducted node classification to measure the learning effectiveness of network representations. In this task, the considered dataset was divided into the training and the test sets. The training set accounted for 60 % and the test set for 40 %. Based on the approach in reference [42], we adopted the logistic regression as the classification method, and used it to train a classification model on the training set. The nodes in the test set were used to test the classification effectiveness. The evaluation metrics included the accuracy (AC), Macro-F1 and Micro-F1, and the results are listed in Table 2.

Method	HEP-TH			BlogCatalog			Epinions		
	AC	Macro	Micro	AC	Macro	Micro	AC	Macro	Micro
Node2Vec	0.481	0.481	0.503	0.698	0.715	0.713	0.175	0.211	0.183
GraphSAGE	0.479	0.452	0.491	0.715	0.724	0.714	0.196	0.228	0.189
GCNs	0.54	0.494	0.523	0.824	0.881	0.881	0.203	0.238	0.202
IproGRU	**0.545**	**0.502**	**0.534**	**0.829**	**0.888**	**0.888**	**0.217**	**0.24**	**0.219**

Table 2. The performance comparison of node classification

In Table 2, IproGRU achieves the best classification results on all three datasets. This good performance can be attributed to two key facts:

1. the influenced nodes merged feature information from their neighboring nodes, and enriched the embedding representation;
2. a more expressive and improved GRU helped the changing network to perform the update operations in time.

5.2 Link Prediction

We conducted link prediction to measure the learning effectiveness of network representations. The methods were trained by extracting a fixed proportion of the edges, and the remaining edges were used as a test set to measure the quality of the prediction in terms of MAP. We deployed IproGRU and three baseline methods (Node2Vec, GraphSAGE, GCNs) on the HEP-TH, BlogCatalog and Epinions datasets for this task, and the results are presented in Table 3.

Method	HEP-TH	BlogCatalog	Epinions
Node2Vec	0.728	0.698	0.605
GraphSAGE	0.773	0.725	0.648
GCNs	0.826	0.769	0.702
IproGRU	**0.857**	**0.783**	**0.747**

Table 3. Comparison of MAP values between IproGRU and three classical methods on link prediction

In Table 3, the MAP of IproGRU is significantly higher than that of GraphSAGE and Node2Vec, and slightly better than that of the well-known GCNs method. The reason is that IproGRU can better learn various rich features on the network and achieve a more accurate network representation. These results demonstrate that enriching the embedding of network nodes is an effective way to improve the learning capability of dynamic network representations.

6 CONCLUSIONS

In this paper, we propose IproGRU, a dynamic network representation learning method based on an improved GRU network that can quickly adapt to the changes occurring in the social network. The embeddings of newly joined unlabeled nodes are quickly generated by aggregating the features neighboring nodes in different layers. In order to track the changes of the influenced nodes in the time series, an improved GRU network is developed to generate more appropriate embeddings for the nodes by enhancing the recent influences of the nodes while appropriately preserving the previously influenced nodes. We conducted node classification and link prediction experiments on three datasets of dynamic graphs. The experimental results show that IproGRU can significantly outperform traditional network representation methods in terms of accuracy and other metrics, which demonstrates the effectiveness of the method in dynamic network representation learning.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61702333, in part by the Opening Topic of the Key Laboratory of Embedded Systems and Service Computing of Ministry of Education under Grant ESSCKF 2019-03, and in part by the Key Innovation Group of Digital Humanities Resource and Research of Shanghai Normal University.

REFERENCES

[1] Xiao, B.—Huang, M.—Barnes, A. J.: Network Closure among Sellers and Buyers in Social Commerce Community. Electronic Commerce Research and Applications, Vol. 14, 2015, No. 6, pp. 641–653, doi: 10.1016/j.elerap.2015.10.001.

[2] Wang, J.—Song, G.—Wu, Y.—Wang, L.: Streaming Graph Neural Networks via Continual Learning. Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), 2020, pp. 1515–1524, doi: 10.1145/3340531.3411963.

[3] Asif, N. A.—Sarker, Y.—Chakrabortty, R. K.—Ryan, M. J.—Ahamed, M. H.—Saha, D. K.—Badal, F. R.—Das, S. K.—Ali, F. M.—Moyeen, S. I.—Islam, M. R.—Tasneem, Z.: Graph Neural Network: A Comprehensive Review on Non-Euclidean Space. IEEE Access, Vol. 9, 2021, pp. 60588–60606, doi: 10.1109/access.2021.3071274.

[4] Mandal, D.—Medya, S.—Uzzi, B.—Aggarwal, C.: MetaLearning with Graph Neural Networks: Methods and Applications. ACM SIGKDD Explorations Newsletter, Vol. 23, 2021, No. 2, pp. 13–22, doi: 10.1145/3510374.3510379.

[5] Qu, M.—Bengio, Y.—Tang, J.: GMNN: Graph Markov Neural Networks. In: Chaudhuri, K., Salakhutdinov, R. (Eds.): Proceedings of the 36th International Conference on Machine Learning (ICML 2019). Proceedings of Machine Learning Research (PMLR), Vol. 97, 2019, pp. 5241–5250, doi: 10.48550/arXiv.1905.06214.

[6] Cao, S.—Lu, W.—Xu, Q.: Deep Neural Networks for Learning Graph Representations. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 30, 2016, No. 1, pp. 1145–1152, doi: 10.1609/aaai.v30i1.10179.

[7] Min, S.—Gao, Z.—Peng, J.—Wang, L.—Qin, K.—Fang, B.: STGSN – A Spatial-Temporal Graph Neural Network Framework for Time-Evolving Social Networks. Knowledge-Based Systems, Vol. 214, 2021, Art. No. 106746, doi: 10.1016/j.knosys.2021.106746.

[8] Jin, D.—You, X.—Li, W.—He, D.—Cui, P.—Fogelman-Soulié, F.—Chakraborty, T.: Incorporating Network Embedding into Markov Random Field for Better Community Detection. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, No. 1, pp. 160–167, doi: 10.1609/aaai.v33i01.3301160.

[9] Bhagat, S.—Cormode, G.—Muthukrishnan, S.: Node Classification in Social Networks. In: Aggarwal, C. (Ed.): Social Network Data Analytics. Springer, Boston, MA, 2011, pp. 115–148, doi: 10.1007/978-1-4419-8462-3_5.

[10] Wang, M.—Qiu, L.—Wang, X.: A Survey on Knowledge Graph Embeddings for Link Prediction. Symmetry, Vol. 13, 2021, No. 3, Art. No. 485, doi: 10.3390/sym13030485.

[11] Berahmand, K.—Nasiri, E.—Rostami, M.—Forouzandeh, S.: A Modified DeepWalk Method for Link Prediction in Attributed Social Network. Computing, Vol. 103, 2021, No. 10, pp. 2227–2249, doi: 10.1007/s00607-021-00982-2.

[12] Su, X.—Xue, S.—Liu, F.—Wu, J.—Yang, J.—Zhou, C.—Hu, W.—Paris, C.—Nepal, S.—Jin, D.—Sheng, Q. Z.—Yu, P. S.: A Comprehensive

Survey on Community Detection with Deep Learning. IEEE Transactions on Neural Networks and Learning Systems, 2022, doi: 10.1109/TNNLS.2021.3137396.

[13] Fan, J.—Sun, Q.—Zhou, W. X.—Zhu, Z.: Principal Component Analysis for Big Data. 2018, doi: 10.48550/arXiv.1801.01602.

[14] de Silva, V.—Tenenbaum, J. B.: Sparse Multidimensional Scaling Using Landmark Points. Technical Report, Stanford University, 2004.

[15] Samko, O.—Marshall, A. D.—Rosin, P. L.: Selection of the Optimal Parameter Value for the Isomap Algorithm. Pattern Recognition Letters, Vol. 27, 2006, No. 9, pp. 968–979, doi: 10.1016/j.patrec.2005.11.017.

[16] Ghojogh, B.—Ghodsi, A.—Karray, F.—Crowley, M.: Locally Linear Embedding and Its Variants: Tutorial and Survey. 2020, doi: 10.48550/arXiv.2011.10925.

[17] Perozzi, B.—Al-Rfou, R.—Skiena, S.: DeepWalk: Online Learning of Social Representations. Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14), 2014, pp. 701–710, doi: 10.1145/2623330.2623732.

[18] Guthrie, D.—Allison, B.—Liu, W.—Guthrie, L.—Wilks, Y.: A Closer Look at Skip-Gram Modelling. Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC '06), ELRA, 2006.

[19] Grover, A.—Leskovec, J.: Node2vec: Scalable Feature Learning for Networks. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16), 2016, pp. 855–864, doi: 10.1145/2939672.2939754.

[20] Tang, J.—Qu, M.—Wang, M.—Zhang, M.—Yan, J.—Mei, Q.: LINE: Large-Scale Information Network Embedding. Proceedings of the 24th International Conference on World Wide Web (WWW '15), ACM, 2015, pp. 1067–1077, doi: 10.1145/2736277.2741093.

[21] Cao, S.—Lu, W.—Xu, Q.: GraRep: Learning Graph Representations with Global Structural Information. Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM '15), 2015, pp. 891–900, doi: 10.1145/2806416.2806512.

[22] Zhu, L.—Guo, D.—Yin, J.—Ver Steeg, G.—Galstyan, A.: Scalable Temporal Latent Space Inference for Link Prediction in Dynamic Social Networks. IEEE Transactions on Knowledge and Data Engineering, Vol. 28, 2016, No. 10, pp. 2765–2777, doi: 10.1109/TKDE.2016.2591009.

[23] Zhou, L.—Yang, Y.—Ren, X.—Wu, F.—Zhuang, Y.: Dynamic Network Embedding by Modeling Triadic Closure Process. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018, No. 1, pp. 571–578, doi: 10.1609/aaai.v32i1.11257.

[24] Hamilton, W. L.—Ying, R.—Leskovec, J.: Inductive Representation Learning on Large Graphs. 2017, doi: 10.48550/arXiv.1706.02216.

[25] Chen, J.—Ma, T.—Xiao, C.: FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. 2018, doi: 10.48550/arXiv.1801.10247.

[26] Nguyen, G. H.—Lee, J. B.—Rossi, R. A.—Ahmed, N. K.—Koh, E.—Kim, S.: Continuous-Time Dynamic Network Embeddings. Companion Proceedings of the The

Web Conference 2018 (WWW '18), 2018, pp. 969-976, doi: 10.1145/3184558.3191526.

[27] Cong, W.—Wu, Y.—Tian, Y.—Gu, M.—Xia, Y.—Mahdavi, M.—Chen, C. C. J.: Dynamic Graph Representation Learning via Graph Transformer Networks. 2021, doi: 10.48550/arXiv.2111.10447.

[28] Ying, C.—Cai, T.—Luo, S.—Zheng, S.—Ke, G.—He, D.—Shen, Y.—Liu, T. Y.: Do Transformers Really Perform Bad for Graph Representation? In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P. S., Wortman Vaughan, J. (Eds.): Advances in Neural Information Processing Systems 34 (NeurIPS 2021). Curran Associates, Inc., 2021, pp. 28877–28888, doi: 10.48550/arXiv.2106.05234.

[29] Vaswani, A.—Shazeer, N.—Parmar, N.—Uszkoreit, J.—Jones, L.—Gomez, A. N.—Kaiser, Ł.—Polosukhin, I.: Attention Is All You Need. In: Guyon, I., Von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.): Advances in Neural Information Processing Systems 30 (NIPS 2017). Curran Associates, Inc., 2017, pp. 5998–6008, doi: 10.48550/arXiv.1706.03762.

[30] Albert, R.—Barabási, A. L.: Statistical Mechanics of Complex Networks. Reviews of Modern Physics, Vol. 74, 2002, No. 1, pp. 47–97, doi: 10.1103/RevModPhys.74.47.

[31] Greff, K.—Srivastava, R. K.—Koutník, J.—Steunebrink, B. R.—Schmidhuber, J.: LSTM: A Search Space Odyssey. IEEE Transactions on Neural Networks and Learning Systems, Vol. 28, 2017, No. 10, pp. 2222–2232, doi: 10.1109/TNNLS.2016.2582924.

[32] Dey, R.—Salem, F. M.: Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks. 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), 2017, pp. 1597–1600, doi: 10.1109/MWSCAS.2017.8053243.

[33] Ma, Y.—Guo, Z.—Ren, Z.—Tang, J.—Yin, D.: Streaming Graph Neural Networks. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), 2020, pp. 719–728, doi: 10.1145/3397271.3401092.

[34] Chung, J.—Gulcehre, C.—Cho, K. H.—Bengio, Y.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. 2014, doi: 10.48550/arXiv.1412.3555.

[35] Gehrke, J.—Ginsparg, P.—Kleinberg, J.: Overview of the 2003 KDD Cup. ACM SIGKDD Explorations Newsletter, Vol. 5, 2003, No. 2, pp. 149–151, doi: 10.1145/980972.980992.

[36] Li, J.—Dani, H.—Hu, X.—Tang, J.—Chang, Y.—Liu, H.: Attributed Network Embedding for Learning in a Dynamic Environment. Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17), 2017, pp. 387–396, doi: 10.1145/3132847.3132919.

[37] Goyal, P.—Kamra, N.—He, X.—Liu, Y.: DynGEM: Deep Embedding Method for Dynamic Graphs. 2018, doi: 10.48550/arXiv.1805.11273.

[38] Zafarani, R.—Liu, H.: Social Computing Data Repository at ASU. 2009.

[39] Kipf, T. N.—Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks. 2016, doi: 10.48550/arXiv.1609.02907.

[40] WANG, D.—CUI, P.—ZHU, W.: Structural Deep Network Embedding. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16), 2016, pp. 1225–1234, doi: 10.1145/2939672.2939753.

[41] GOYAL, P.—CHHETRI, S. R.—CANEDO, A.: Dyngraph2vec: Capturing Network Dynamics Using Dynamic Graph Representation Learning. Knowledge-Based Systems, Vol. 187, 2020, Art. No. 104816, doi: 10.1016/j.knosys.2019.06.024.

[42] MAHDAVI, S.—KHOSHRAFTAR, S.—AN, A.: Dynnode2vec: Scalable Dynamic Network Embedding. 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 3762–3765, doi: 10.1109/bigdata.2018.8621910.

Jianguo Pan received his Ph.D. degree from the College of Computer Science and Technology, Shanghai University, Shanghai, China, in 2009. He is currently Associate Professor in the College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai, China. His current research interests include artificial intelligence and Internet of Things.

Huan Li received her bachelor's degree from the College of Computer Science and Technology, Anhui University of Science and Technology, Anhui, China, in 2016. She is currently pursuing her master's degree at the College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai, China. Her research interests include data mining and big data.

Jiajun Teng received her M.Sc. degree at the College of Information and Mechanical, Electrical Engineering, Shanghai Normal University, Shanghai, China, in 2021. Her research interests are data mining and big data.

Qin Zhao received his B.Sc. degree from Shanghai Ocean University, Shanghai, China, in 2004, and his M.Sc. and Ph.D. degrees from the Tongji University, Shanghai, in 2009 and 2016, respectively. He is currently Associate Professor with the College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai. His research interests include data mining, big data, social networking, and machine learning.

Maozhen Li received his Ph.D. from the Institute of Software, Chinese Academy of Sciences in 1997. He was Post-Doctoral Scholar in the School of Computer Science and Informatics, Cardiff University, UK in 1999–2002. He is currently Professor in the School of Electronic and Computer Engineering at Brunel University, UK. His research interests are in the areas of high-performance computing (grid and cloud computing) for big data analysis and intelligent systems. He has over 100 research publications in these areas. He is Fellow of the British Computer Society.