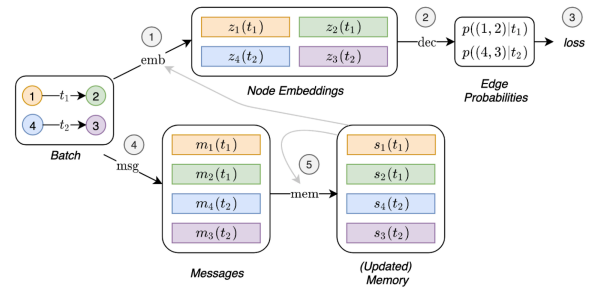
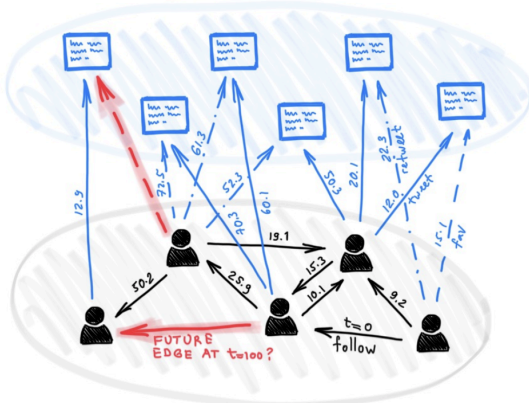

TGN: Temporal Graph Networks [arXiv, YouTube, Blog Post]

Dynamic Graph

TGN



Introduction

Despite the plethora of different models for deep learning on graphs, few approaches have been proposed thus far for dealing with graphs that present some sort of dynamic nature (e.g. evolving features or connectivity over time).

In this paper, we present Temporal Graph Networks (TGNs), a generic, efficient framework for deep learning on dynamic graphs represented as sequences of timed events. Thanks to a novel combination of memory modules and graph-based operators, TGNs are able to significantly outperform previous approaches being at the same time more computationally efficient.

We furthermore show that several previous models for learning on dynamic graphs can be cast as specific instances of our framework. We perform a detailed ablation study of different components of our framework and devise the best configuration that achieves state-of-the-art performance on several transductive and inductive prediction tasks for dynamic graphs.

Paper link: Temporal Graph Networks for Deep Learning on Dynamic Graphs

Running the experiments

Requirements

Dependencies (with python ≥ 3.7):

```
{bash} pandas==1.1.0 torch==1.6.0 scikit_learn==0.23.1
```

Dataset and Preprocessing

Download the public data Download the sample datasets (eg. wikipedia and reddit) from here and store their csv files in a folder named `data/`.

Preprocess the data We use the dense `numpy` format to save the features in binary format. If edge features or nodes features are absent, they will be replaced by a vector of zeros.

```
{bash} python utils/preprocess_data.py --data wikipedia --bipartite python utils/preprocess_data.py --data reddit --bipartite
```

Model Training

Self-supervised learning using the link prediction task: “

```
{bash} # TGN-attn: Supervised learning on the wikipedia dataset python train_self_supervised.py --use_memory --prefix tgn-attn --n_runs 10
```

TGN-attn-reddit: Supervised learning on the reddit dataset

```
python train_self_supervised.py -d reddit --use_memory --prefix tgn-attn-reddit --n_runs 10
```

```
1
2 Supervised learning on dynamic node classification (this requires a
   trained model from
3 the self-supervised task, by eg. running the commands above):
4 ```{bash}
5 # TGN-attn: self-supervised learning on the wikipedia dataset
6 python train_supervised.py --use_memory --prefix tgn-attn --n_runs 10
7
8 # TGN-attn-reddit: self-supervised learning on the reddit dataset
9 python train_supervised.py -d reddit --use_memory --prefix tgn-attn-
   reddit --n_runs 10
```

Baselines

```
““{bash} ### Wikipedia Self-supervised
```

Jodie

```
python train_self_supervised.py -use_memory -memory_updater rnn -embedding_module time -  
prefix jodie_rnn -n_runs 10
```

DyRep

```
python train_self_supervised.py -use_memory -memory_updater rnn -dyrep -use_destination_embedding_in_mes  
-prefix dyrep_rnn -n_runs 10
```

Reddit Self-supervised

Jodie

```
python train_self_supervised.py -d reddit -use_memory -memory_updater rnn -embedding_module  
time -prefix jodie_rnn_reddit -n_runs 10
```

DyRep

```
python train_self_supervised.py -d reddit -use_memory -memory_updater rnn -dyrep -use_destination_embedding  
-prefix dyrep_rnn_reddit -n_runs 10
```

Wikipedia Supervised

Jodie

```
python train_supervised.py -use_memory -memory_updater rnn -embedding_module time -prefix  
jodie_rnn -n_runs 10
```

DyRep

```
python train_supervised.py -use_memory -memory_updater rnn -dyrep -use_destination_embedding_in_message  
-prefix dyrep_rnn -n_runs 10
```

Reddit Supervised

Jodie

```
python train_supervised.py -d reddit -use_memory -memory_updater rnn -embedding_module  
time -prefix jodie_rnn_reddit -n_runs 10
```

DyRep

```
python train_supervised.py -d reddit -use_memory -memory_updater rnn -dyrep -use_destination_embedding_in_  
-prefix dyrep_rnn_reddit -n_runs 10
```

```
1  
2  
3 ### Ablation Study  
4 Commands to replicate all results in the ablation study over different  
5   modules:  
6   ```{bash}  
7   # TGN-2l  
8   python train_self_supervised.py --use_memory --n_layer 2 --prefix tgn-2  
9     l --n_runs 10  
10  
11  # TGN-no-mem  
12  python train_self_supervised.py --prefix tgn-no-mem --n_runs 10  
13  
14  # TGN-time  
15  python train_self_supervised.py --use_memory --embedding_module time --  
16    prefix tgn-time --n_runs 10  
17  
18  # TGN-id  
19  python train_self_supervised.py --use_memory --embedding_module  
20    identity --prefix tgn-id --n_runs 10  
21  
22  # TGN-sum  
23  python train_self_supervised.py --use_memory --embedding_module  
24    graph_sum --prefix tgn-sum --n_runs 10  
25  
26  # TGN-mean  
27  python train_self_supervised.py --use_memory --aggregator mean --prefix  
28    tgn-mean --n_runs 10
```

General flags {txt} optional arguments: -d DATA, --data DATA Data
sources to use (wikipedia or reddit)--bs BS Batch size --prefix
PREFIX Prefix to name checkpoints and results --n_degree N_DEGREE

Number of neighbors to sample at each layer --n_head N_HEAD Number of heads used in the attention layer --n_epoch N_EPOCH Number of epochs --n_layer N_LAYER Number of graph attention layers --lr LR Learning rate --patience Patience of the early stopping strategy --n_runs Number of runs (compute mean and std of results) --drop_out DROP_OUT Dropout probability --gpu GPU Idx **for** the gpu to use --node_dim NODE_DIM Dimensions of the node embedding --time_dim TIME_DIM Dimensions of the time embedding --use_memory Whether to use a memory **for** the nodes --embedding_module Type of the embedding module --message_function Type of the message function --memory_updater Type of the memory updater --aggregator Type of the message aggregator --memory_update_at_the_end Whether to update the memory at the end or at the start of the batch --message_dim Dimension of the messages --memory_dim Dimension of the memory --backprop_every Number of batches to process before performing backpropagation --different_new_nodes Whether to use different unseen nodes **for** validation and testing --uniform Whether to sample the temporal neighbors uniformly (or instead take the most recent ones) --randomize_features Whether to randomize node features --dyrep Whether to run the model as DyRep

TODOs

- Make code memory efficient: for the sake of simplicity, the memory module of the TGN model is implemented as a parameter (so that it is stored and loaded together of the model). However, this does not need to be the case, and more efficient implementations which treat the models as just tensors (in the same way as the input features) would be more amenable to large graphs.

Cite us

```
1 @inproceedings{tgn_icml_grl2020,  
2   title={Temporal Graph Networks for Deep Learning on Dynamic Graphs  
3     },  
4   author={Emanuele Rossi and Ben Chamberlain and Fabrizio Frasca and  
5     Davide Eynard and Federico  
6     Monti and Michael Bronstein},  
7   booktitle={ICML 2020 Workshop on Graph Representation Learning},  
8   year={2020}
```