Continuous-Time Dynamic Network Embeddings

By Nguyen et al.

EGOR DMITRIEV, Utrecht Univeristy, The Netherlands

1 GOALS

- Describe a general framework for incorporating temporal information into network embedding methods
- Methods for learning time-respecting embeddings from continuous-time dynamic networks
- TLDR: Describes a temporal walk strategy

2 PRELIMINARIES

- …

3 CHALLENGES

- **General & Unifying Framework**: general framework for incorporating temporal dependencies in node embedding and deep graph models that leverage random walks
- **Continuous-Time Dynamic Networks**: timedependent network representation for continuous-time dynamic networks
- **Effectiveness**: Must outperform baselines

4 PREVIOUS WORK / CITATIONS

- Static Snapshot Graphs:
  - each static snapshot graph represents all edges that occur between a user-specified discrete-time interval (e.g., day or week)
  - Refs: [57, 59, 63, 64]
  - Drawbacks:
    * Noisy approximation on continuous time
    * Selecting appropriate granularity
- **This Work:**
  - Random Walks
  - Supports **graph streams** (edges come and go live)
  - Any work using random walks can benefit from the proposed methods

5 DEFINITIONS

- **Continuous-Time Dynamic Network**: $G = (V, E_T, \mathcal{T})$
  - E_T edges at continuous times (actually events)
- **Temporal Walk**: temporal walk represents a temporally valid sequence of edges traversed in increasing order of edge times
- **Temporal Neighborhood**: $\Gamma_t(v) = \{(w, t') \mid e = (v, w, t') \in E_T \land \mathcal{T}(e) > t\}$
  - Neighbors of a node v at time t
  - Nodes may appear multiple times (multiple edge events)
- –

6 OUTLINE / STRUCTURE

- Random Walks

  - – Changes walk space $\mathbb{S}$ to $\mathbb{S}_T$
- **Goal**: $f : V \to \mathbb{R}^D$: mapping nodes in G to D-dimensional **time-dependent feature representation**
  - – For ml tasks such as link prediction
- **Temporal Walks**:
  - – Require starting time (Randomly samples or from a randomly samples edge)
  - – Edges from further time may be less predictive (so bias wisely)
  - – Has min length ω
- **Biasing the walks**
  - – Unbiased: $Pr(e) = 1/N$
  - – Biased: Used a distribution based on time
  - – Favor newer edges: $Pr(e) = \frac{\exp[\mathcal{T}(e) - t_{\min}]}{\sum_{e' \in E_T} \exp[\mathcal{T}(e') - t_{\min}]}$
    - ∗ Exp dist with $t_m in$ as starting time
- **Biasing Neighbor selection**: Uniform or Biased (bias for time difference for example)
  - – Walk bias van be reused based on $\tau(v)$
- Temporal Context Windows:
  - – Window count: $\beta = \sum_{i=1}^{k} |S_{t_i}| - \omega + 1$
    - ∗ Number of walks that can be derived from the window with size ω
- node2vec approach for

Algorithm 1 Continuous-Time Dynamic Network Embeddings

Input:
    a (un)weighted and (un)directed dynamic network $G = (V, E_T, \mathcal{T})$,
    temporal context window count β, context window size ω,
    embedding dimensions D,

1 Set maximum walk length $L = 80$
2 Initialize set of *temporal walks* S_T to $\emptyset$
3 Initialize number of context windows $C = 0$
4 Precompute sampling distribution $\mathbb{F}_s$ using G
        $\mathbb{F}_s \in \{\text{Uniform, Exponential, Linear}\}$
5 $G' = (V, E_T, \mathcal{T}, \mathbb{F}_s)$
6 **while** $\beta - C > 0$ **do**
7   Sample an edge $e_* = (v, u)$ via distribution $\mathbb{F}_s$
8     $t = \mathcal{T}(e_*)$
9     $S_t = $ TEMPORALWALK$(G', e_* = (v, u), t, L, \omega + \beta - C - 1)$
10   **if** $|S_t| > \omega$ **then**
11   Add the *temporal walk* S_t to S_T
12      $C = C + (|S_t| - \omega + 1)$
13 **end while**
14 $\mathbf{Z} = $ STOCHASTICGRADIENTDESCENT(ω, D, S_T)
15 **return** the *dynamic* node embedding matrix $\mathbf{Z}$

Algorithm 2 Temporal Random Walk

1 **procedure** TEMPORALWALK$(G', e = (s, r), t, L, C)$
2   Initialize temporal walk $S_t = [\, s, r \,]$
3   Set $i = r$ ▷ current node
4   **for** $p = 1$ to $min(L, C) - 1$ **do**
5      $\Gamma_t(i) = \{(w, t') \mid e = (i, w, t') \in E_T \wedge \mathcal{T}(i) > t\}$
6   **if** $|\Gamma_t(i)| > 0$ **then**
7   Select node j from distribution $\mathbb{F}_\Gamma(\Gamma_t(i))$
8   Append j to S_t
9   Set $t = \mathcal{T}(i, j)$
10   Set $i = j$
11   **else** terminate temporal walk
12   **return** temporal walk S_t of length $|S_t|$ rooted at node s

---

7 EVALUATION

- Try different versions of network (2 main components to swap)
- Baselines: node2vec [26], DeepWalk [52], and LINE [65]. ,
- Datasets from NetworkRepository [58].
-

8 CODE

- …

9 RESOURCES

- …