DynGEM: Deep Embedding Method for Dynamic Graphs

Palash Goyal, Nitin Kamra[1], Xinran He, Yan Liu

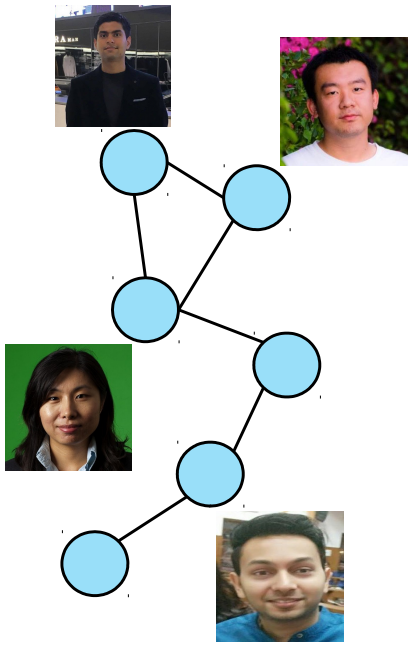
Department of Computer Science
University of Southern California

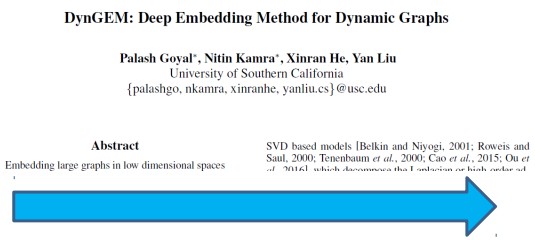
August 14, 2017

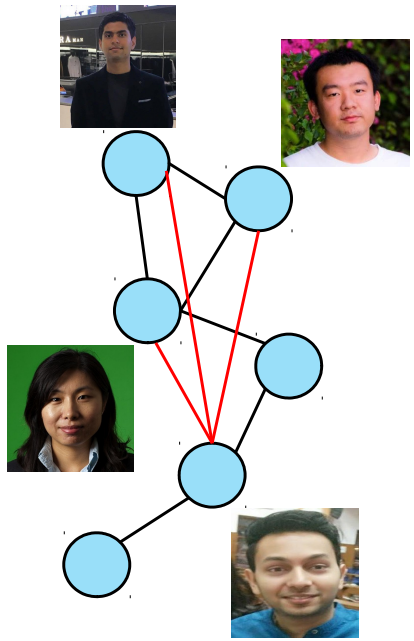
[1] Nitin Kamra and Palash Goyal have equal contribution

Outline

Dynamic Graphs



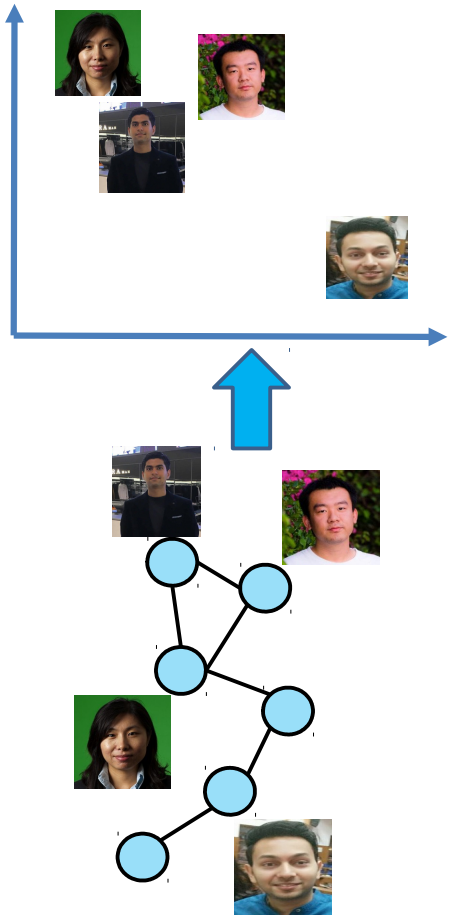




Definition

Real world graphs evolve by addition and deletion of nodes and edges.
We represent a dynamic graph as a snapshot of static graphs.

Dynamic Graph Embedding



DynGEM: Deep Embedding Method for Dynamic Graphs

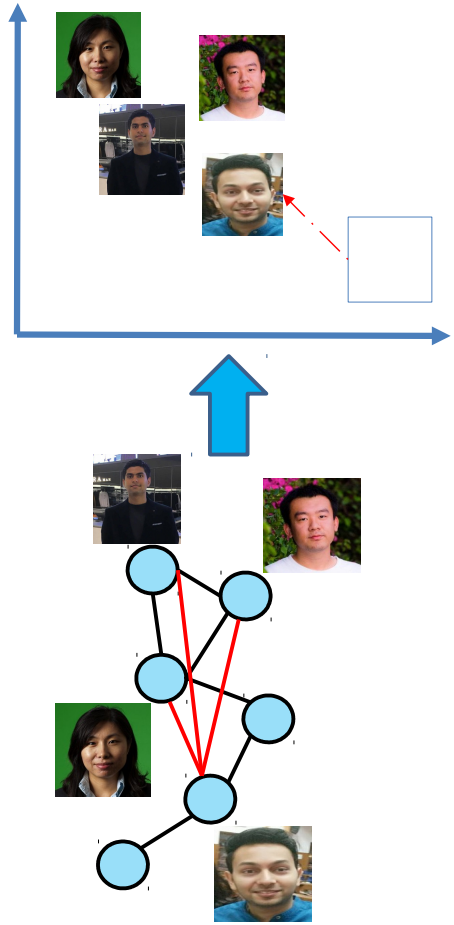
Palash Goyal*, Nitin Kamra*, Xinran He, Yan Liu
University of Southern California
{palashgo, nkamra, xinranhe, yanliu.cs}@usc.edu

Abstract

Embedding large graphs in low dimensional spaces

SVD based models [Belkin and Niyogi, 2001; Roweis and Saul, 2000; Tenenbaum *et al.*, 2000; Cao *et al.*, 2015; Ou *et*

DynGEM paper accepted



Problem Statement

Graph Embedding

It aims to represent each vertex of a graph in a low dimensional space $\mathbb{R}^d$ while preserving certain properties of a graph, i.e., it learns the function $f_G : V \rightarrow \mathbb{R}^d$, where $d \ll |V|$.

Problem Statement

Graph Embedding

It aims to represent each vertex of a graph in a low dimensional space $\mathbb{R}^d$ while preserving certain properties of a graph, i.e., it learns the function $f_G : V \rightarrow \mathbb{R}^d$, where $d \ll |V|$.

Dynamic Graph Embedding

It extends the concept of embedding to dynamic graphs. Given a dynamic graph G, a dynamic graph embedding is a time series of mappings $F = \{f_1, \ldots, f_T\}$ such that mapping f_t is a graph embedding for G_t and all mappings preserve the proximity measure for their respective graphs.

Possible Solutions and Challenges

- Apply static embedding algorithm at each time step
  - Problem: Non-unique embedding

Possible Solutions and Challenges

- Apply static embedding algorithm at each time step
  - Problem: Non-unique embedding
- Realign embeddings at consecutive time steps
  - May work for factorization based models
  - Problem: Linear alignment will give sub-par performance on non-linear embedding approaches

Embedding Stability

$$\mathcal{S}_{rel}(\mathcal{F};t) = \frac{\|F_{t+1}(V_t) - F_t(V_t)\|_F}{\|F_t(V_t)\|_F} \Big/ \frac{\|S_{t+1}(V_t) - S_t(V_t)\|_F}{\|S_t(V_t)\|_F}$$

- Relative change in embedding
- Relative change in graph

$$K_{\mathcal{S}}(\mathcal{F}) = \max_{\tau,\tau'} |\mathcal{S}_{rel}(F;\tau) - \mathcal{S}_{rel}(F;\tau')|.$$

Embedding Stability

$$\mathcal{S}_{rel}(\mathcal{F};t) = \frac{\|F_{t+1}(V_t) - F_t(V_t)\|_F}{\|F_t(V_t)\|_F} \Big/ \frac{\|S_{t+1}(V_t) - S_t(V_t)\|_F}{\|S_t(V_t)\|_F}$$

- Relative change in embedding
- Relative change in graph

$$K_{\mathcal{S}}(\mathcal{F}) = \max_{\tau,\tau'} |\mathcal{S}_{rel}(F;\tau) - \mathcal{S}_{rel}(F;\tau')|.$$

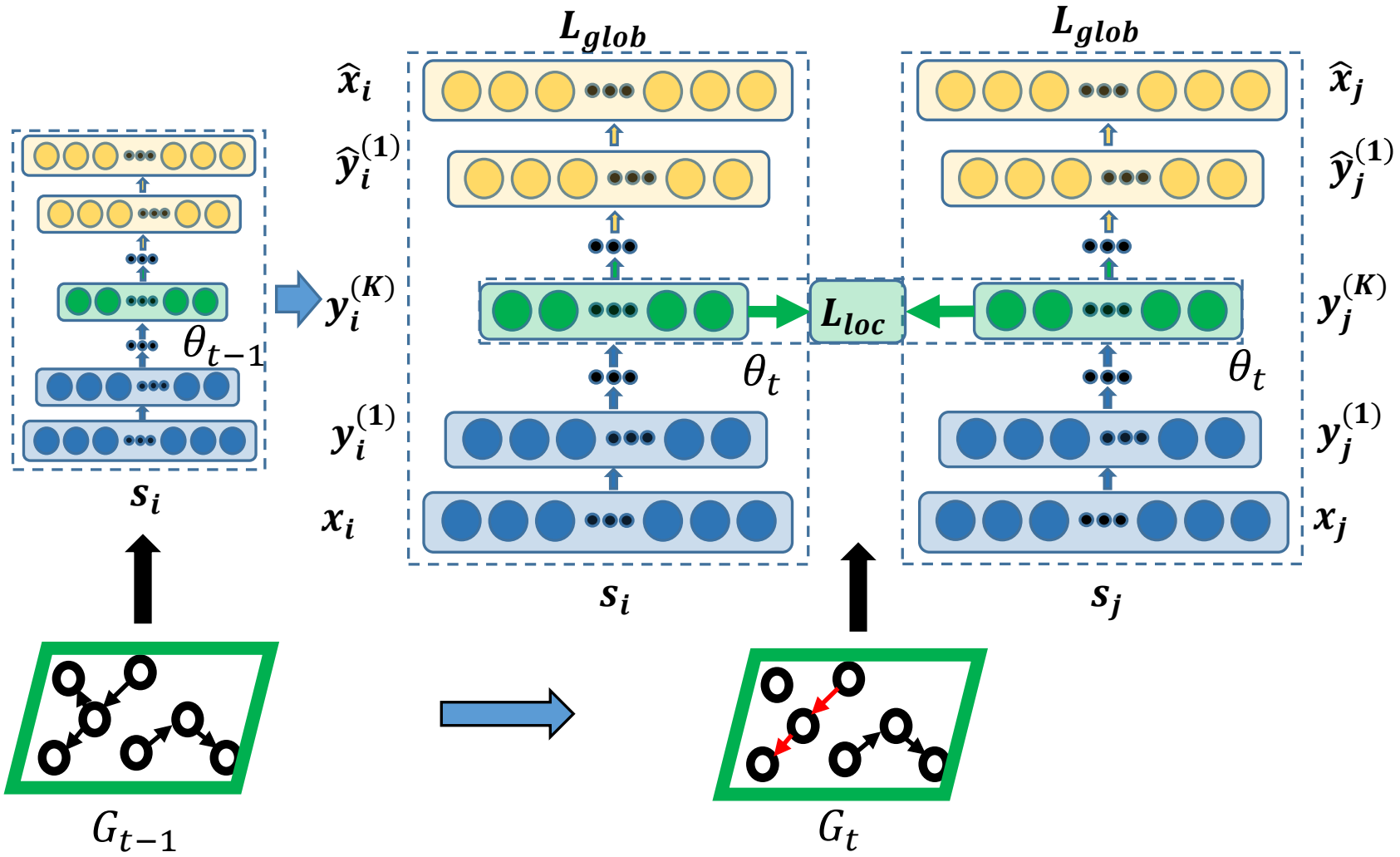
Embedding Stability

$$\mathcal{S}_{rel}(\mathcal{F};t) = \frac{\|F_{t+1}(V_t) - F_t(V_t)\|_F}{\|F_t(V_t)\|_F} \Big/ \frac{\|S_{t+1}(V_t) - S_t(V_t)\|_F}{\|S_t(V_t)\|_F}$$

- Relative change in embedding
- Relative change in graph

$$K_{\mathcal{S}}(\mathcal{F}) = \max_{\tau,\tau'} |\mathcal{S}_{rel}(F;\tau) - \mathcal{S}_{rel}(F;\tau')|.$$

Dynamic Graph Embedding Model (DynGEM)



Handling Growing Graphs

- Addition of nodes in the graph may require additional model parameters
- Get width hidden layers using *PropSize* heuristic
  - $size(l_{k+1}) \geq \rho \times size(l_k)$
- Deepen the model if *PropSize* is not satisfied for embedding layer
- Adopt $Net2WiderNet$ and $Net2DeeperNet$ to expand the autoencoder

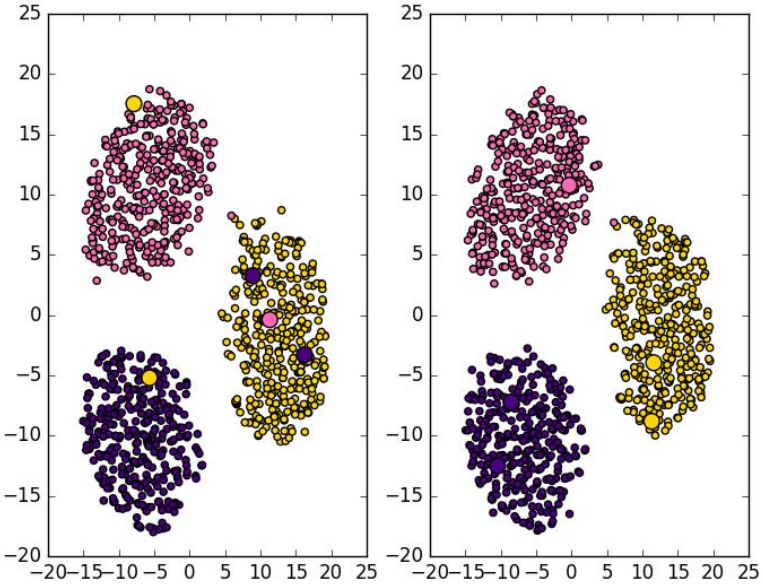
Algorithm 1: Algorithm: DynGEM

Input: $G_t = (V_t, E_t), G_{t-1} = (V_{t-1}, E_{t-1})$, Y_{t-1}, $\boldsymbol{\theta}_{t-1}$
Output: Embedding Y_t
From G_t, generate adjacency matrix S, penalty matrix B
Create the autoencoder model with initial architecture
Initialize $\boldsymbol{\theta}_t$ randomly **if** $t = 1$, **else** $\boldsymbol{\theta_t} = \boldsymbol{\theta}_{t-1}$
if $|V_t| > |V_{t-1}|$ **then**
    Compute new layer sizes with PropSize heuristic
    Expand autoencoder layers and/or insert new layers
end if
Create set $\mathcal{S} = \{(s_i, s_j)\}$ for each $e = (v_i, v_j) \in E_t$
for $i = 1, 2, \ldots$ **do**
    Sample a minibatch M from $\mathcal{S}$
    Compute gradients $\nabla_{\boldsymbol{\theta}_t} L_{net}$ of objective L_{net} on M
    Do gradient update on $\boldsymbol{\theta}_t$ with nesterov momentum
end for

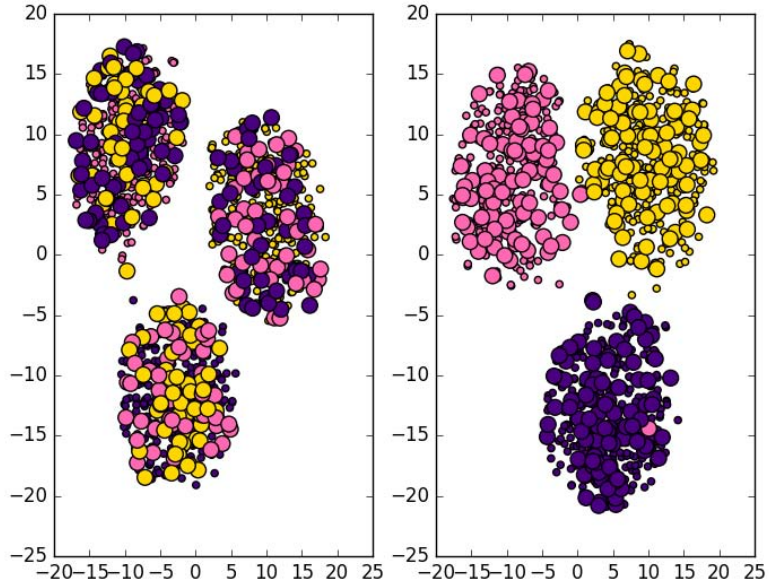
Datasets

- **Synthetic Data (SYN)**
  - Generated using Stochastic Block Model
  - 1000 nodes, 79,800-79,910 edges
- **High Energy Physics (HEP-TH)**
  - Author collaboration network
  - 1,424-7,980 nodes, 2,556-21,036 edges
- **Autonomous Systems (AS)**
  - Router communication network
  - 7716 nodes, 10,695-26,46 edges
- **Enron (ENRON)**
  - Email network
  - 184 nodes, 63-591 edges

Visualization



(a) DynGEM time step with 5 nodes jumping out of 1000



(b) DynGEM time step with 300 nodes jumping out of 1000

Graph Reconstruction

- Reconstruct graph edges using decoder
- Rank pairs of vertices according to reconstructed proximity

	SYN	HEP-TH	AS	ENRON
GF_{align}	0.119	0.49	0.164	0.223
GF_{init}	0.126	**0.52**	0.164	0.31
$SDNE_{align}$	0.124	0.04	0.07	0.141
SDNE	**0.987**	0.51	0.214	0.38
DynGEM	**0.987**	0.491	**0.216**	**0.424**

Table: Average MAP of graph reconstruction.

Link Prediction

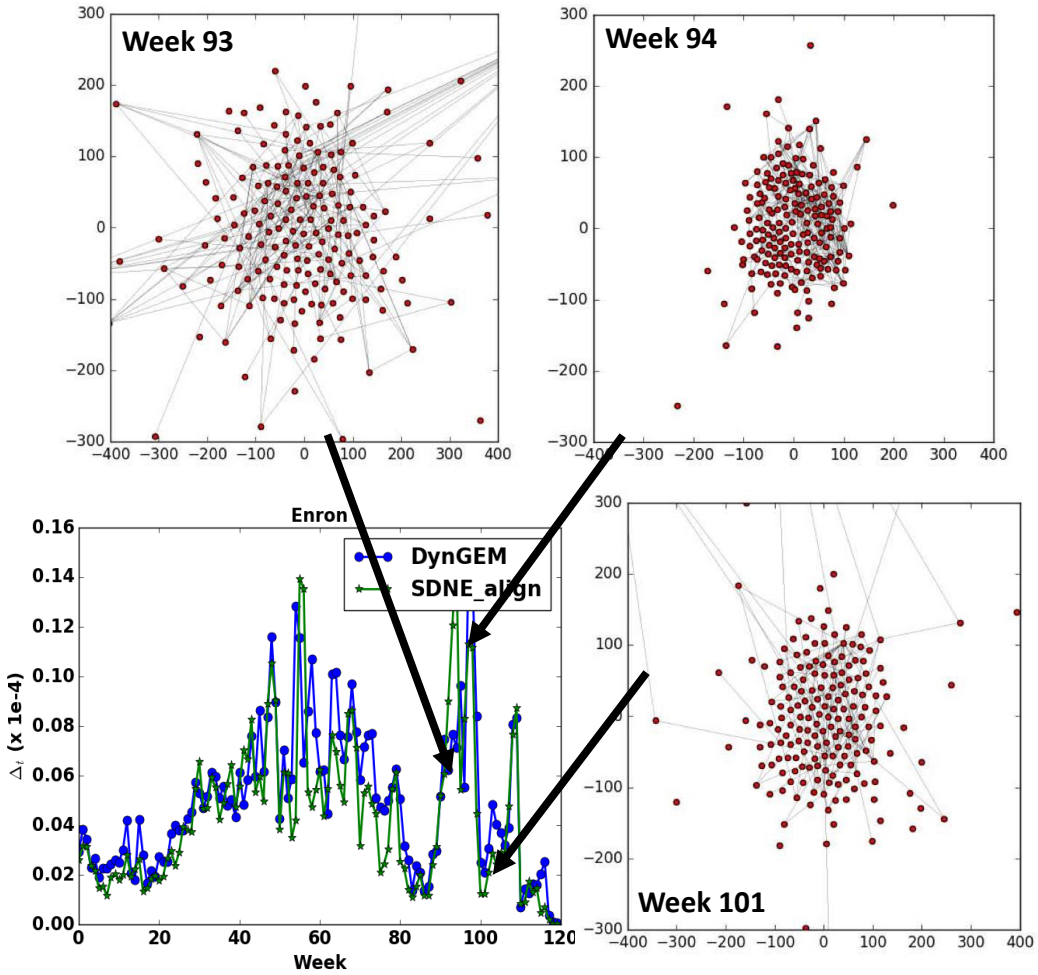
- Randomly hide 15% of network edges at time t
- Train the model using graph snapshots till time t
- Test the prediction using hidden edges

	SYN	HEP-TH	AS	ENRON
GF_{align}	0.027	0.04	0.09	0.021
GF_{init}	0.024	0.042	0.08	0.017
$SDNE_{align}$	0.031	0.17	0.1	0.06
SDNE	0.034	0.1	0.09	0.081
DynGEM	**0.194**	**0.26**	**0.21**	**0.084**

Table: Average MAP of link prediction.

Anomaly Detection

- Detected anomalies in Enron by thresholding norm of change in consecutive embeddings



Stability and Scalability

- Computed stability constant and speedup achieved

	SYN	HEP-TH	AS	ENRON
SDNE	0.18	14.715	6.25	19.722
SDNE$_{align}$	0.11	8.516	2.269	18.941
DynGEM	**0.008**	**1.469**	**0.125**	**1.279**

Table: Stability constants $K_{\mathcal{S}}(F)$ of embeddings on dynamic graphs.

	SYN	HEP-TH	AS	ENRON
SDNE$_{align}$	56.6 min	71.4 min	210 min	7.69 min
DynGEM	13.8 min	25.4 min	80.2 min	3.48 min
Speedup	4.1	2.81	2.62	2.21
Speedup$_{exp}$	4.8	3	3	3

Table: Computation time of embedding methods.

Conclusion

- Developed a model that can embed a dynamic graph in vector space
- Introduced the concept of stability in dynamic graph embedding
- Extended the model to handle growing graphs
- Experiments on graph reconstruction, link prediction, stability, scalability and anomaly detection show benefits over existing approaches

Questions?