Dynamic Network Representation Learning via Gaussian Embedding

Abstract

Network embedding plays a principal role in graph mining. However, two major limitations exist in previous network embedding methods: dynamics modeling and uncertainty modeling. In this paper, we propose a dynamic network embedding method with Gaussian Embedding, *DNGE*, to overcome these limitations. *DNGE* learns node representations for dynamic networks in the space of Gaussian distributions and models dynamic information by integrating temporal smoothness as the regularization. Experiments in community detection and link prediction demonstrate the effectiveness of our method.

1 Introduction

Network data is ubiquitous in daily life and analyzing networks is of both theoretical and practical values. Recent years have witnessed numerous approaches to analyze networks, e.g., community detection [1], role discovery [2], and link prediction [3]. As relations exist between nodes that disobey the *i.i.d* assumption, it is non-trivial to apply traditional data mining techniques in networks directly. Network embedding fills this gap by mapping nodes in a network into a low-dimensional space according to their structural information. Many attempts showed the effectiveness of using network embedding techniques in different network analysis tasks [4, 5, 6, 7].

To learn effective network representations, different models have been developed in recent years but there are some limitations. Early network embedding methods mainly focus on static networks [6, 7, 5]. However, many real-world networks are dynamic with evolving structures. The existence of dynamics makes network analysis a more challenging problem. To solve this problem, recently some studies have been proposed to take into account the temporal information to learn embeddings, either in dynamic networks [8, 9, 10] or streaming networks [11, 12, 13]. Nonetheless, real-world networks, especially dynamic and temporal networks, may be noisy and incomplete and in most cases these uncertainties are inevitable because of anomaly or missing information. Thus, another limitation still exists: how to capture the uncertainties of embeddings. Most existing point vector representations are deterministic [14] and are not capable of modeling the uncertainties of node representations.

Motivated by these observations, we tackle the two major challenges in learning network representations: *dynamics modeling* and *uncertainty modeling*. In this paper we propose a dynamic network embedding method with Gaussian Embedding, *DNGE*. *DNGE* learns node representations for dynamic networks in the space of Gaussian distributions and models dynamic information by integrating temporal smoothness as the regularization. Thus, it can capture temporal information and model uncertainties in dynamic networks. We propose two different strategies, i.e., smoothness in means and smoothness in distributions, to model the dynamic information. Smoothness in means guarantees the learned node representations are consistent and smoothness in distributions allows that both embeddings and uncertainties can be shared between two consecutive network snapshots. To evaluate the performance of the proposed *DNGE*, we conduct community detection and link prediction experiments on both synthetic and real-world networks. The results demonstrate the effectiveness of our method in structure preservation, dynamics modeling and uncertainty modeling.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

2 The Proposed Model

A dynamic network is defined as $\mathcal{G} = \{G^t|t = 1, ..., T\}$ which consists of a series of graph snapshots $G^t = \{V, E^t\}$. We assume edges to be undirected for simplicity and we also assume that the nodes are fixed in all snapshots and the edges change over time.

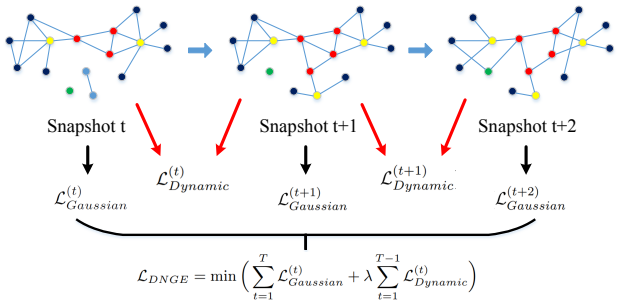

Figure 1: Graphical representation of *DNGE* framework.

Definition 1 *Given a dynamic network $\mathcal{G} = \{G^t|t = 1, ..., T\}$, the problem of **Dynamic Network Gaussian Embedding** aims to represent each node $v_i \in V$ in each snapshot t into a Gaussian distribution $P_i^{(t)}$ with mean $\mu_i^{(t)}$ and covariance $\Sigma_i^{(t)}$ in a low-dimensional space $\mathbb{R}^d$, i.e., learning a function $f : V \to \mathcal{N}(x; \mu, \Sigma)$, where $\mu \in \mathbb{R}^d$ is the mean, $\Sigma \in \mathbb{R}^{d\times d}$ is the covariance and $d \ll |V|$. In the space $\mathbb{R}^d$, the temporal information of nodes is preserved using explicit temporal regularization and the uncertainty of node representations is captured by Σ.*

To solve the problem of Dynamic Network Gaussian Embedding by overcoming the limitations of *dynamics modeling* and *uncertainty modeling*, we propose a dynamic network embedding framework with Gaussian embedding (DNGE). *DNGE* consists of two components: Gaussian embedding component which learns node representations and models uncertainties, and dynamics modeling component which captures temporal information and smooths learned node representations. Formally, the objective function is defined as:

$$\mathcal{L}_{DNGE} = \min\left(\sum_{t=1}^{T}\mathcal{L}^{(t)}_{Gaussian} + \lambda\sum_{t=1}^{T-1}\mathcal{L}^{(t)}_{Dynamic}\right), \tag{1}$$

where $\mathcal{L}^{(t)}_{Gaussian}$ is the Gaussian embedding component in t^{th} snapshot and $\mathcal{L}^{(t)}_{Dynamic}$ is the dynamics modeling component between snapshot t and $t + 1$. λ controls the importance of dynamics and in an extreme case, *DNGE* is equal to the static Gaussian embedding when $\lambda = 0$. The graphical representation of *DNGE* is shown in Fig. 1.

2.1 Gaussian Embedding Component

Gaussian embedding component maps each node i in the graph into a Gaussian distribution P_i with mean μ_i and covariance Σ_i. The objective function of Gaussian embedding is defined as:

$$\mathcal{L}_{Gaussian} = -\sum_{(v,u)\in\Gamma_+}\mathcal{E}(P_v, P_u) + \sum_{(v',u')\in\Gamma_-}\mathcal{E}(P_{v'}, P_{u'}), \tag{2}$$

where Γ_+ and Γ_- are the positive and negative pairs, respectively. $\mathcal{E}(\cdot,\cdot)$ is the energy function to measure the similarity between two distributions, P_v and P_u are the learned Gaussian distributions for nodes v and u. To reduce the computational complexity, for each Gaussian distribution $P \sim \mathcal{N}(x; \mu, \Sigma)$, we only consider the diagonal covariance for Σ. The key idea of network embedding is that two nodes connected in the network should have similar representations, i.e., similar Gaussian distributions in Gaussian embedding. Therefore, we should measure the similarity between two Gaussian distributions. We use KL divergence based energy function to measure the similarity between two Gaussian distributions in this study. Formally the KL divergence based energy function $\mathcal{E}_{KL}(P_i, P_j)$ is defined as:

$$\begin{aligned}\mathcal{E}_{KL}(P_i, P_j) &= D_{KL}(P_i, P_j) \\ &= \int_{x\in\mathbb{R}}\mathcal{N}(x; \mu_i, \Sigma_i)\log\frac{\mathcal{N}(x; \mu_j, \Sigma_j)}{\mathcal{N}(x; \mu_i, \Sigma_i)}dx = \frac{1}{2}\Big\{tr(\Sigma_i^{-1}\Sigma_j) + \mu^T\Sigma_i^{-1}\mu - \log\frac{\det(\Sigma_j)}{\det(\Sigma_i)} - d\Big\},\end{aligned} \tag{3}$$

where $\mu = \mu_i - \mu_j$, d is the number of dimensions, P_i and P_j are the learned Gaussian embeddings for node i and j respectively. $tr(\cdot)$ is the trace function, $\det(\cdot)$ denotes the determinant of a matrix.

Table 1: Clustering performance on Epinions network.

Method		Purity	NMI
static embedding methods	DeepWalk	0.1842	0.1456
	LINE	0.1322	0.1239
	node2vec	0.1713	0.1360
dynamic embedding methods	Dyn Triad	0.1840	**0.1629**
	DynGEM	0.1594	0.1441
	DANE	0.1425	0.1216
Gaussian embedding	Gauss Emb	0.1865	0.1503
	$DNGE_{Mean}$	0.1873	0.1526
	$DNGE_{Dist}$	**0.1942**	0.1615

In this work, we apply the one-order proximity hypothesis same to [6, 14], i.e., two connected nodes should have similar representations. Thus, the positive pair set Γ_+ (in Eq. (2)) are the edge set $E = \{(i,j)|i,j \in V\}$ and the negative pair set Γ_- are generated from random sampling.

2.2 Dynamics Modeling Component

Dynamics modeling component can capture the temporal information and smooth the learned node representations. To achieve this goal, we propose two smoothness strategies as a regularization term, smoothness in *means* and smoothness in *distributions*.

Smoothness in means guarantees the learned node representations, i.e., means of Gaussian distributions, are consistent. The most straightforward way to achieve it is to constrain the change of means in two consecutive network snapshots. Therefore, the formal definition of smoothness in *means* is defined as the square of Frobenius norm of means as follows: $SM_i^{(t)} = \|\mu_i^{(t)} - \mu_i^{(t-1)}\|_F^2$.

Smoothness in distributions considers both means and covariances and allows that both embeddings and uncertainties can be shared between two consecutive snapshots. Formally, it is defined as: $SD_i^{(t)} = \mathcal{E}(P_i^{(t)}, P_i^{(t-1)})$, where $\mathcal{E}(\cdot,\cdot)$ is the method to measure the similarity between two distributions. To keep consistent, we continue to use KL divergence shown in Eq. (3).

To combine the Gaussian embedding component (Eq. (2)) and dynamics modeling component (smoothness in means or distributions), we have the objective function for DNGE. For smoothness in means, in t^{th} snapshot we define the objective function $\mathcal{L}_{DNGE-Mean}^{(t)}$ as:

$$\min \sum_{\Gamma_+^{(t)}} \mathcal{E}_{KL}(P_i^{(t)}, P_j^{(t)}) - \sum_{\Gamma_-^{(t)}} \mathcal{E}_{KL}(P_{i'}^{(t)}, P_{j'}^{(t)}) + \lambda \sum_{\Gamma^{(t)}} (SM_i^{(t)} + SM_j^{(t)}), \tag{4}$$

where $\Gamma^{(t)}$ is obtained by concatenating positive pairs $\Gamma_+^{(t)}$ and negative pairs $\Gamma_-^{(t)}$ in snapshot t. Similarly, for smoothness in distributions, in t^{th} snapshot the objective function $\mathcal{L}_{DNGE-Dist}^{(t)}$ is:

$$\min \sum_{\Gamma_+^{(t)}} \mathcal{E}_{KL}(P_i^{(t)}, P_j^{(t)}) - \sum_{\Gamma_-^{(t)}} \mathcal{E}_{KL}(P_{i'}^{(t)}, P_{j'}^{(t)}) + \lambda \sum_{\Gamma^{(t)}} (SD_i^{(t)} + SD_j^{(t)}), \tag{5}$$

In this supplementary material, we discuss the retails on model learning.

3 Experimental Analysis

Setup and Datasets To validate the effectiveness of *DNGE*, we conduct community detection and link prediction experiments to evaluate the performance of *DNGE* on real-world networks. For community detection, we compare the average clustering performance over all snapshots. For link prediction, we learn node embeddings on the first $T-1$ network snapshots and predict the links on the last snapshot. Similar to [5], we regard link prediction as a classification task: edges as positive examples and randomly selected node pairs from each snapshot which have no edge connecting them as negative examples. Without loss of generality, learned node embeddings are used as the features and logistic regression is used as the classifier. Normalized Mutual Information (NMI) and Purity are employed to verify the community results. AUC is used as the evaluation metric in link prediction. Additionally, we analyze the uncertainty modeling results using synthetic networks.

Table 2: Link prediction results of different methods.

Data set		Enron	Messages	Reality	Facebook
Methods		AUC			
static embedding embedding	DeepWalk	0.7564	0.6007	0.5874	0.5457
	LINE	0.7535	0.5213	0.7018	0.5564
	node2vec	0.7819	0.6687	0.5910	0.5384
dynamic embedding embedding	Dyn Triad	0.8355	0.8504	**0.7204**	0.7255
	DynGEM	0.8129	0.8297	0.6776	0.7004
	DANE	0.8023	0.7912	0.6743	0.7122
Gaussian embedding	Gauss Emb	0.7883	0.8116	0.6317	0.6213
	$DNGE_{Mean}$	0.8130	0.8458	0.6814	0.6221
	$DNGE_{Dist}$	**0.8373**	**0.8967**	0.7109	**0.7422**

Table 3: Clustering performance on synthetic network.

Methods	Number of noisy edges					
	0	1000	2000	3000	4000	5000
node2vec	0.9090	0.8693	0.8612	0.8441	0.8412	0.8223
Gauss Emb	0.8825	0.8254	0.8104	0.8002	0.8124	0.7991
Dyn Triad	0.9472	0.8743	0.8722	0.8589	0.8603	0.8502
$DNGE_{Mean}$	0.9287	0.8655	0.8743	0.8552	0.8522	0.8475
$DNGE_{Dist}$	0.9533	0.9101	0.9032	0.8699	0.8653	0.8701

Baselines. We compare 3 types of baseline methods: (1) Static embedding methods: methods map nodes to deterministic vectors in static networks including DeepWalk [6], LINE [7] and *node2vec* [5]. (2) Dynamic embedding methods: Dynamic Triad (Dyn Triad) [10], DynGEM [15] and DANE [16] are compared. (3) Gaussian embedding methods: we compare Gauss Emb [17] and our proposed *DNGE* using two dynamics modeling strategies, i.e., $DNGE_{Mean}$ and $DNGE_{Dist}$.

Results. The average NMI and Purity for community detection on the Epinions network are shown in Table 1. From these results, it can be observed that *DNGE* with smoothness in distributions outperforms other baselines on Purity and performs the second best on NMI. The results demonstrate *DNGE* can effectively preserve network structures. Smoothness in distributions is better than that in mean and this is because smoothness in distributions can learn more consistent representations.

The AUC scores on the real-world networks are shown in Table 2. From these results, we can draw the following conclusions: (1) *DNGE* outperforms other baselines on most networks. The strategy of smoothness in distributions performs better than smoothness in means because it models both embeddings and uncertainties as the dynamic regularization. This result is same to that in community detection. (2) In general, Gaussian embedding methods achieve better performance than point embedding methods. It demonstrates that Gaussian embedding is a better method for representation learning in network data. In fact, Gaussian embedding can provide better interpretations of embeddings from the perspective of probability and Bayesian statistics.

It is intuitive that the more noisy edges a node has, the less discriminative information it contains, thus making its embedding more uncertain. We synthesize a dynamic network consisting of 400 nodes in 4 snapshots and these nodes belong to 4 communities. To introduce noise, we add different numbers of random edges to each snapshots from 1000 to 5000. The results of different embedding methods are shown in Table 3. Note that we select some representative methods from all baselines. From these results, it can be observed that our *DNGE* can effectively learn embeddings in noisy networks. *DNGE* with distribution smoothness is better than mean smoothness.

4 Conclusions

We proposed a novel dynamic network embedding framework using Gaussian embedding *DNGE* to tackle the limitations of *dynamics modeling* and *uncertainty modeling*. *DNGE* learns node representations by modeling temporal information as regularization. Furthermore, *DNGE* utilizes Gaussian embedding to represent each node as a Gaussian distribution to model uncertainties. Experimental study demonstrated that *DNGE* effectively preserves community structures, captures dynamic information, and models uncertainties.

References

[1] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[2] Ryan A Rossi and Nesreen K Ahmed. Role discovery in networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):1112–1131, 2015.

[3] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology*, 58(7):1019–1031, 2007.

[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 891–900. ACM, 2015.

[5] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[6] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[7] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[8] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. 2019.

[9] Lun Du, Yun Wang, Guojie Song, Zhicong Lu, and Junshan Wang. Dynamic network embedding: An extended approach for skip-gram based network embedding. In *IJCAI*, pages 2086–2092, 2018.

[10] Le-kui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *AAAI*, 2018.

[11] Stephen Bonner, John Brennan, Ibad Kureshi, Georgios Theodoropoulos, Andrew Stephen McGough, and Boguslaw Obara. Temporal graph offset reconstruction: Towards temporally robust graph representation learning. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3737–3746. IEEE, 2018.

[12] Giang H Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Dynamic network embeddings: From random walks to temporal random walks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1085–1092. IEEE, 2018.

[13] Xi Liu, Ping-Chun Hsieh, Nick Duffield, Rui Chen, Muhe Xie, and Xidao Wen. Real-time streaming graph embedding through local actions. 2019.

[14] Ludovic Dos Santos, Benjamin Piwowarski, and Patrick Gallinari. Multilabel classification on heterogeneous graphs with gaussian embeddings. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 606–622. Springer, 2016.

[15] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*, 2018.

[16] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. Attributed network embedding for learning in a dynamic environment. *arXiv preprint arXiv:1706.01860*, 2017.

[17] Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623*, 2014.

Appendix A Model Learning

To combine the Gaussian embedding component (Eq. (2)) and dynamics modeling component (Eq. (**??**) and Eq. (**??**)), we have the objective function for DNGE. For smoothness in means, in t^{th} snapshot we define the objective function $\mathcal{L}_{DNGE}^{(t)}$ as:

$$\min \sum_{\Gamma_+^{(t)}} \mathcal{E}_{KL}(P_i^{(t)}, P_j^{(t)}) - \sum_{\Gamma_-^{(t)}} \mathcal{E}_{KL}(P_{i'}^{(t)}, P_{j'}^{(t)}) + \lambda \sum_{\Gamma^{(t)}} (SM_i^{(t)} + SM_j^{(t)}), \tag{6}$$

where $\Gamma^{(t)}$ is obtained by concatenating positive pairs $\Gamma_+^{(t)}$ and negative pairs $\Gamma_-^{(t)}$ in snapshot t. We can compute the gradients of this objective function with respect to the means $\mu^{(t)}$ and covariances $\Sigma^{(t)}$:

$$\begin{aligned}
\frac{\partial \mathcal{L}^{(t)}}{\partial \mu_i^{(t)}} &= -\Delta_{ij}^{(t)} + \lambda(2\mu_i^{(t)} - \mu_i^{(t-1)} - \mu_i^{(t+1)}) \\
\frac{\partial \mathcal{L}^{(t)}}{\partial \mu_j^{(t)}} &= \Delta_{ij}^{(t)} + \lambda(2\mu_j^{(t)} - \mu_j^{(t-1)} - \mu_j^{(t+1)}) \\
\frac{\partial \mathcal{L}^{(t)}}{\partial \Sigma_i^{(t)}} &= \frac{1}{2}\Big((\Sigma_i^{(t)})^{-1}\Sigma_j^{(t)}(\Sigma_i^{(t)})^{-1} + \Delta_{ij}^{(t)}\Delta_{ij}^{(t)T} - (\Sigma_i^{(t)})^{-1}\Big) \\
\frac{\partial \mathcal{L}^{(t)}}{\partial \Sigma_j^{(t)}} &= \frac{1}{2}\Big((\Sigma_j^{(t)})^{-1} - (\Sigma_i^{(t)})^{-1}\Big)
\end{aligned} \tag{7}$$

where $\Delta_{ij}^{(t)} = (\Sigma_i^{(t)})^{-1}(\mu_i^{(t)} - \mu_j^{(t)})$. Note that for boundary conditions when $t = 1$ and $t = T$, some terms in the gradient will vary slightly.

Similarly, for smoothness in distributions, in t^{th} snapshot we define the objective function $\mathcal{L}_{DNGE}^{(t)}$ as:

$$\min \sum_{\Gamma_+^{(t)}} \mathcal{E}_{KL}(P_i^{(t)}, P_j^{(t)}) - \sum_{\Gamma_-^{(t)}} \mathcal{E}_{KL}(P_{i'}^{(t)}, P_{j'}^{(t)}) + \lambda \sum_{\Gamma^{(t)}} (SD_i^{(t)} + SD_j^{(t)}), \tag{8}$$

The gradients for means $\mu^{(t)}$ and covariances $\Sigma^{(t)}$ are:

$$\begin{aligned}
\frac{\partial \mathcal{L}^{(t)}}{\partial \mu_i^{(t)}} &= -\Delta_{ij}^{(t)} - \lambda(\tilde{\Delta}_i^{(t)} + \tilde{\Delta}_i^{(t+1)}) \\
\frac{\partial \mathcal{L}^{(t)}}{\partial \mu_j^{(t)}} &= \Delta_{ij}^{(t)} - \lambda(\tilde{\Delta}_j^{(t)} + \tilde{\Delta}_j^{(t+1)}) \\
\frac{\partial \mathcal{L}^{(t)}}{\partial \Sigma_i^{(t)}} &= \frac{1}{2}\Big\{\Big((\Sigma_i^{(t)})^{-1}\Sigma_j^{(t)}(\Sigma_i^{(t)})^{-1} + \Delta_{ij}^{(t)}\Delta_{ij}^{(t)T} - (\Sigma_i^{(t)})^{-1}\Big) \\
&+ \Big((\Sigma_i^{(t)})^{-1} - (\Sigma_i^{(t+1)})^{-1}\Big) + \Big((\Sigma_i^{(t)})^{-1}(\Sigma_i^{(t-1)})(\Sigma_i^{(t)})^{-1} + \tilde{\Delta}_i^{(t)}\tilde{\Delta}_i^{(t)T} - (\Sigma_i^{(t)})^{-1}\Big)\Big\} \\
\frac{\partial \mathcal{L}^{(t)}}{\partial \Sigma_j^{(t)}} &= \frac{1}{2}\Big\{\Big((\Sigma_j^{(t)})^{-1} - (\Sigma_i^{(t)})^{-1}\Big) + \Big((\Sigma_j^{(t)})^{-1} - (\Sigma_j^{(t+1)})^{-1}\Big) \\
&+ \Big((\Sigma_j^{(t)})^{-1}(\Sigma_j^{(t-1)})(\Sigma_j^{(t)})^{-1} + \tilde{\Delta}_j^{(t)}\tilde{\Delta}_j^{(t)T} - (\Sigma_j^{(t)})^{-1}\Big)\Big\}
\end{aligned} \tag{9}$$

where $\Delta_{ij}^{(t)} = (\Sigma_i^{(t)})^{-1}(\mu_i^{(t)} - \mu_j^{(t)})$, $\tilde{\Delta}_i^{(t)} = (\Sigma_i^{(t)})^{-1}(\mu_i^{(t)} - \mu_i^{(t-1)})$ and $\tilde{\Delta}_j^{(t)} = (\Sigma_j^{(t)})^{-1}(\mu_j^{(t)} - \mu_j^{(t-1)})$.

Note that to avoid the means to grow to large and the covariances to be positive definite as well as reasonably sized, we regularize the means and covariances to learn the embedding. Due to the different geometric characteristics, two different hard constraint strategies have been used for means μ_i and covariances Σ_i, respectively. In particular, we have

$$\begin{aligned}
\|\mu_i\| &\leq C, \ \forall i \\
c_{min} I \prec \Sigma_i &\prec c_{max} I, \ \forall i,
\end{aligned} \tag{10}$$

where C, c_{min} and c_{max} are the constraint parameters.

We use AdaGrad to optimize the parameters. The learning procedure is described in Algorithm 1. Initialization phase is from line 1 to 5, context generation is shown from line 6 to 8, and Gaussian embeddings are learned from line 9 to 12.

Algorithm 1 The Learning Algorithm of *DNGE*

Input: A dynamic graph $\mathcal{G} = \{G^t | t = 1, ..., T\}$, embedding dimension d, constraint values c_{max} and c_{min} for covariance, learning rate α, and max number of iterations n.

Output: Gaussian embeddings (mean vector $\mu_v^{(t)}$ and covariance matrix $\Sigma_v^{(t)}$) for nodes $v \in V$ in snapshot t

for all $v \in V$ in each snapshot t **do**
  Initialize mean $\mu_v^{(t)}$ for v in snapshot t
  Initialize covariance $\Sigma_v^{(t)}$ for v in snapshot t
  Regularize $\mu_v^{(t)}$ and $\Sigma_v^{(t)}$ with constraint in Eq. (10)
end for
for all snapshot t **do**
  Generate positive and negative sets $\Gamma_+^{(t)}$ and $\Gamma_-^{(t)}$ for each node
end for
while not reach the maximum iteration n **do**
  Update means and covariances based on Eq. (7) or (9)
  Regularize μ and Σ with constraint in Eq. (10)
end while

Table 4: A brief statistics of real-world networks.

Data set	# nodes	# edges	# snapshots
Epinions	8518	300548	10
Enron	147	1666	9
Messages	1889	59835	5
Reality	6809	9467	10
Facebook	44416	196414	12

Appendix B Experiment Settings

We conduct experiments on five real-world networks from different domains. The first data set, i.e., Epinions, is used for community detection[1]. The remaining data sets are used for link prediction[2]. A brief overview of these networks is shown in Table 4.

Mean and covariance constrain parameters C, c_{min} and c_{max} are set to be 1.0, 0.5, and 2.0 respectively in all experiments. In community detection experiments, the latent dimension is 100 for all embedding methods. For DeepWalk and *node2vec*, the number of walks is 10, walk length is 80 and window size is 10. For *node2vec*, $p = 1$ and $q = 0.5$. For Dynamic Triad and DANE, we use the default settings. In link prediction experiments, the latent dimension of Enron, Message, Reality and Facebook are set to be 32, 64, 100 and 100, respectively. Other settings are same to community detection.

[1] `https://www.cse.msu.edu/~tangjili/trust.html`

[2] `http://networkrepository.com/dynamic.php`