# A Generative Model for Dynamic Networks with Applications

**Shubham Gupta, Gaurav Sharma, Ambedkar Dukkipati**
Department of Computer Science and Automation
Indian Institute of Science
Bangalore - 560012, India
email: {shubhamg, ambedkar}@iisc.ac.in

## Abstract

Networks observed in real world like social networks, collaboration networks etc., exhibit temporal dynamics, i.e. nodes and edges appear and/or disappear over time. In this paper, we propose a generative, latent space based, statistical model for such networks (called dynamic networks). We consider the case where the number of nodes is fixed, but the presence of edges can vary over time. Our model allows the number of communities in the network to be different at different time steps. We use a neural network based methodology to perform approximate inference in the proposed model and its simplified version. Experiments done on synthetic and real world networks for the task of community detection and link prediction demonstrate the utility and effectiveness of our model as compared to other similar existing approaches.

## 1 Introduction

Many networks encountered in real world evolve with time, i.e. new nodes and edges appear while some existing ones disappear. For example, consider the network of people connected via Facebook, where the users and their interactions change over time. It is interesting to study the underlying dynamics of such networks to understand what drives these changes, how communities are formed, how will a network behave in future and what has lead to the current state of the network etc.

In static network setting, one of the problems that has been extensively studied is community detection (Fortunato 2010). In a dynamic setting, this problem becomes more challenging since communities themselves take birth, meet death, grow, shrink, split and merge etc. (Rossetti and Cazabet 2017). In this paper, we address the problem of modeling dynamic networks that exhibit a community structure where the number of nodes is invariant over time but the presence of edges is time dependent.

A wide range of dynamic networks with different characteristics can be found in real world. In order to model these networks in a meaningful way, it is essential to focus on certain specific types of networks. In this paper, we consider undirected (edges are not directional), assortative (similar types of nodes have a high probability of connecting), positive influence based (adjacent nodes positively influence a

given node) and gradually evolving networks. Many real world networks such as friendship networks, communication networks etc., fall in this category.

We propose a generative model, which we call Evolving Latent Space Model (ELSM), for dynamic networks of the type mentioned above. The main advantage of our model over existing models (Xing, Fu, and Song 2010; Foulds et al. 2011; Heaukulani and Ghahramani 2013; Kim and Leskovec 2013; Xu and Hero 2014), is that our model uses a more flexible, continuous latent space and we allow the number of communities to vary over time (also supported by (Kim and Leskovec 2013)). It also allows one to generate a dynamic graph with a temporally *stable* community structure. The purpose of proposing this generative model is three-folds: (i) it will help in understanding how networks evolve over time, (ii) it will provide synthetic data for other dynamic community detection and link prediction algorithms to operate on and (iii) it will allow model based inference for problems like community detection and link prediction.

Though ELSM can be used both as a generative model and an inference model, for the inference problem, we propose a simplified version of our model that is computationally less expensive. This model can be used exclusively for inference and we refer to it as iELSM. Inference in ELSM is given in supplementary material (Gupta, Sharma, and Dukkipati 2018). Exact inference in ELSM and iELSM is intractable, thus we resort to variational techniques for performing approximate inference. We use a recurrent neural network (RNN) to model various variational parameters. We also present an extension of our model that deals with weighted graphs in the experiments section (§5).

Our main contributions are: **(i)** We propose a generative model (ELSM) for dynamic networks with fixed number of nodes that allows the number of communities to vary over time. **(ii)** We derive a variational approximation to exact inference in ELSM and iELSM and use a neural network architecture to learn variational parameters. **(iii)** We outline a general inference methodology that can be customized in various ways by using problem specific information. **(iv)** We demonstrate the applicability of proposed model and inference methodology by performing experiments on real and synthetic networks for community detection and link prediction where we obtain state of the art results.

## 2 Evolving Latent Space Model

**Notation:** We use capital bold letters ($\mathbf{A}$, $\mathbf{\Sigma}$ etc.) for matrices and set of vectors, small bold letters ($\mathbf{m}$, $\boldsymbol{\alpha}$ etc.) for vectors and small letters ($a$, $b$ etc.) for scalars. If a quantity is associated with time step $t$, we add $(t)$ to the superscript as in $\mathbf{A}^{(t)}$. We use the same letter for denoting set of vectors and an element in this set. For example, $\mathbf{\Sigma}^{(t)}$ is a set of vectors associated with time step $t$ and $\boldsymbol{\sigma}_i^{(t)}$ is the $i^{th}$ vector in this set. The $j^{th}$ element of a vector $\mathbf{c}$ is denoted by $c_j$ (note that this is not bold).

Our model draws motivation from two phenomena that are frequently observed in real world networks: (i) *similar* nodes are more likely to connect to each other and (ii) over a time, nodes tend to become similar to their neighbors. We aim to model a latent space, in which the euclidean distance between embeddings of any two nodes is inversely proportional to their similarity. In this space, operations mimicking the two phenomena listed above can be modeled naturally as we explain later in this section.

To make the underlying motivations concrete, our running example in this section would pertain to the political inclinations of $n$ hypothetical individuals. In this context, the latent space can be thought of as an ideological space where the axes correspond to different political ideologies. Each individual can be embedded in this space based on their inclination towards different ideologies. Distance between two latent embeddings will then correspond to difference in political standpoints of the corresponding individuals. The observations will be symmetric, binary adjacency matrices $\{\mathbf{A}^{(t)}\}_{t=1}^T$ where $a_{ij}^{(t)} = 1$ if individuals $i$ and $j$ interacted with each other in time window $t$. One can see that the two phenomena listed in the previous paragraph naturally occur in this setting, i.e., individuals tend to interact with people who share their ideology and over time the peer group of an individual positively influences their ideology.

Our proposed generative model can be summarized as follows: first, a set of initial latent embeddings, $\mathbf{Z}^{(1)} = \{\mathbf{z}_1^{(1)}, \mathbf{z}_2^{(1)}, ..., \mathbf{z}_n^{(1)}\}$, is created. Based on the latent embeddings $\mathbf{Z}^{(1)}$ the observed adjacency matrix $\mathbf{A}^{(1)}$ is sampled. For $t = 2, 3 ... T$, latent embeddings at time $t$, $\mathbf{Z}^{(t)}$, are updated based on $\mathbf{Z}^{(t-1)}$, $\mathbf{A}^{(t-1)}$ and some extra information about possible new communities at time-step $t$ and $\mathbf{A}^{(t)}$ is sampled based on $\mathbf{Z}^{(t)}$. Next, we describe each component of this process in detail.

### 2.1 Getting the Initial Set of Embeddings

Let $K$ denote the maximum number of clusters at time $t = 0$. We take $K$ and a multinomial distribution over $K$ elements, $\pi$, as input. A vector $\mathbf{c} \in \mathbb{R}^n$ is created such that it assigns an integer between 1 and $K$ sampled from $\pi$ to each node . In the general case, we sample the initial community centers $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_K \sim \mathcal{N}(\mathbf{m}, s^2\mathbf{I})$, where $\mathbf{m} \in \mathbb{R}^d$ and $s^2 \in \mathbb{R}$ are hyperparameters and $d$ is the dimension of latent space. The embedding for each node in the first snapshot can then be sampled as:

$$\mathbf{z}_i^{(1)} \sim \mathcal{N}(\boldsymbol{\mu}_{c_i}, s_1^2\mathbf{I}), \quad i = 1, 2, ..., n \qquad (1)$$
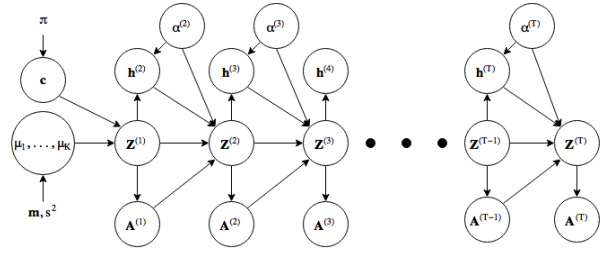


Figure 1: Graphical Model for ELSM. $h_i^{(t)} = 1$ for nodes that join the newly created community with center $\boldsymbol{\alpha}^{(t)}$ and 0 otherwise. This *splitting* mechanism models the birth of new communities.

The parameter $s_1$ dictates the *spread* within communities. In the context of our example, this step corresponds to the creation of an initial set of political ideologies and individuals subscribing to these ideologies. Since $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_K$ are sampled independently, there will be at-most $K$ distinct communities in latent space for the first time step. To encode prior information about communities in the model, rather than sampling $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_K$ independently from a normal distribution, one can specify these vectors explicitly.

### 2.2 Generating Observed $\mathbf{A}^{(t)}$ from Latent $\mathbf{Z}^{(t)}$

In ELSM, the probability of an edge between two nodes is inversely proportional to the distance between their embeddings in the latent space. By this, one hopes to get embeddings that are easier to deal with for a clustering algorithm like *k-means*. We model this as follows:

$$a_{ij}^{(t)} = a_{ji}^{(t)} \sim \text{Bernoulli}\Big(f(\mathbf{z}_i^{(t)} - \mathbf{z}_j^{(t)})\Big), \quad i > j, \quad (2)$$

where $f(.)$ is a nonlinear function. The observed matrices are symmetric and we do not allow self loops. In our experiments on generation of synthetic networks, we used $f(x) = 1 - \tanh(||x||_2^2/s_2^2)$, where $s_2$ is a parameter that controls the radius around a node's embedding in the latent space within which it is more likely to connect to other nodes.

We will demonstrate via experiments in §5, that one can have other choices for the function $f(.)$ and the distribution parameterized by $f(.)$ depending on problem context. For example, in the case of weighted graph with positive integer weights, Poisson distribution can be used.

### 2.3 Emergence of New Communities

If the nodes are only allowed to become similar to their neighbors, then over time the latent embeddings will collapse onto a single point and all the community structure in the network will be lost. But such a phenomenon is not observed in real world networks. In the absence of any other guiding force, to avoid the collapse of latent space and to model the emergence of new communities ELSM randomly generates a new community center at each time step.

At each time step $t$, a new community center $\boldsymbol{\alpha}^{(t)}$ is sampled from the same prior that was used in the first layer to

generate the original community centers $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_K$, i.e. $\boldsymbol{\alpha}^{(t)} \sim \mathcal{N}(\mathbf{m}, s^2\mathbf{I})$. We define a Bernoulli random variable $h_i^{(t)}$ for each node $i$ at time $t$. If $h_i^{(t)} = 1$, then node $i$'s updated latent embedding is sampled as:

$$\mathbf{z}_i^{(t)} \sim \mathcal{N}(\boldsymbol{\alpha}^{(t)}, s_1^2\mathbf{I}) \tag{3}$$

The parameter $s_1$ has the same meaning as in (1). If $h_i^{(t)} = 0$, then node $i$'s latent embedding will be updated based on the embeddings of its neighbors at time step $t-1$ by the process described in the next subsection. We model the probability $P(h_i^{(t)} = 1)$ as a function of the distance between $\boldsymbol{\alpha}^{(t)}$ and $\mathbf{z}_i^{(t-1)}$. If the latent embedding of node $i$ at time step $t-1$ is close to $\boldsymbol{\alpha}^{(t)}$, then $P(h_i^{(t)} = 1)$ will be high and vice-versa. We model this as follows:

$$h_i^{(t)} \sim \text{Bernoulli}\Big(g(\mathbf{z}_i^{(t-1)} - \boldsymbol{\alpha}^{(t)})\Big), \tag{4}$$

where $g(.)$ is a nonlinear function. In our experiments for generating synthetic data, we use $g(x) = 1 - \tanh(||x||_2^2/s_3^2)$. The parameter $s_3$ controls the influence of the newly created community on other nodes. Note that this step does not necessarily involve the creation of a new community as the sampled $\boldsymbol{\alpha}^{(t)}$ may lie within an existing community in the latent space.

In the context of our example, this step will correspond to new political ideologies appearing in the latent space. Individuals that have a similar political ideology are more likely to embrace this new ideology. Individuals from different communities may come together and form a new political ideology of their own.

## 2.4 Evolving the Latent Space

ELSM tries to make nodes similar to their neighbors over time. To model this, for time step $t = 2, 3, ..., T$, a mean vector $\boldsymbol{\mu}_i^{(t)}$ is obtained for each node $i$ as follows:

$$\begin{aligned}
\boldsymbol{\mu}_i^{(t)} = & \frac{1}{1 + \sum_{j \neq i} a_{ij}^{(t-1)} l(\mathbf{z}_i^{(t-1)} - \mathbf{z}_j^{(t-1)})} \times \Big(\mathbf{z}_i^{(t-1)} \\
& + \sum_{j \neq i} a_{ij}^{(t-1)} l(\mathbf{z}_i^{(t-1)} - \mathbf{z}_j^{(t-1)}) \mathbf{z}_j^{(t-1)}\Big)
\end{aligned} \tag{5}$$

Here, we use $l(x) = e^{-||x||^2/s_4^2}$ and $s_4$ is a parameter that controls the influence of neighbors on a node. Note that only the immediate neighbors at time $t-1$ can affect the future embedding of a node at time $t$. Also, note that the value of $\boldsymbol{\mu}_i^{(t)}$ lies in the convex hull formed by the embeddings of the node and its neighbors in the latent space a time step $t-1$. If $h_i^{(t)} = 0$ for a given node, then the updated embedding of the node is given by:

$$\mathbf{z}_i^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_i^{(t)}, s_1^2\mathbf{I}) \tag{6}$$

Then, (3) and (6) can be combined to get the general update equation for the evolution of latent embeddings of the nodes as:

$$\mathbf{z}_i^{(t)} \sim \mathcal{N}(h_i^{(t)}\boldsymbol{\alpha}^{(t)} + (1 - h_i^{(t)})\boldsymbol{\mu}_i^{(t)}, s_1^2\mathbf{I}), \quad i = 1, 2, ..., n \tag{7}$$

**Algorithm 1** Generating Synthetic Networks

**Input:** $n, T, \pi, K, \mathbf{m}, s, s_1, s_2, s_3, s_4$
Sample $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_K \sim \mathcal{N}(\mathbf{m}, s^2\mathbf{I})$
Sample $c_1, c_2, ..., c_n \sim \pi$
Sample $\mathbf{Z}^{(1)}$ using (1)
Sample $\mathbf{A}^{(1)}$ using (2)
**for** $t = 2$ **to** $T$ **do**
    Sample $\boldsymbol{\alpha}^{(t)} \sim \mathcal{N}(\mathbf{m}, s^2\mathbf{I})$
    Sample $h^{(t)}$ using (4)
    Sample $\mathbf{Z}^{(t)}$ using (7)
    Sample $\mathbf{A}^{(t)}$ using (2)
**end for**
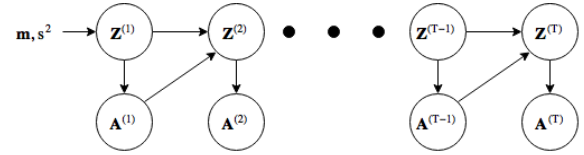**Return:** $\{\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, ..., \mathbf{A}^{(T)}\}$



Figure 2: Graphical Model for iELSM

This not only allows the nodes to become similar over time, it also allows them to move apart to form new communities as it is observed in real world networks. In the context of our example, (5) says that an individual's political ideology will be influenced by the political ideology of their peers.

It is easy to see that ELSM supports operations like birth, death, growth, shrinkage, merge and split on the communities. The latent space offers a nice community structure (as we will show via our experiments, visualization of latent space is in supplementary material) because the operations presented in (2), (4) and (5) depend on distances between embeddings in the latent space. Figure 1 presents the underlying graphical model for ELSM. The procedure for generating synthetic networks is listed in Algorithm 1.

# 3 Inference Network

Inference is concerned with finding the latent embedding of each node at each time step, $\{\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)}, ..., \mathbf{Z}^{(T)}\}$, that best explains an observed dynamic network specified by the sequence of adjacency matrices $\{\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, ..., \mathbf{A}^{(T)}\}$. In this section, we describe the inference procedure for a simplified version of ELSM. We argue that this simplified version, which we call Evolving Latent Space Model for Inference (iELSM), is as good as the original ELSM for the task of inference. Inference for ELSM follows along the same lines as iELSM and full details have been worked out in supplementary material.

## 3.1 Evolving Latent Space Model for Inference

While generating a synthetic dynamic network using ELSM, a guiding force is needed to: (i) select the initial community membership for each node so that there is a community structure in the first layer and (ii) force nodes to move

apart over time and form new communities. In the absence of any other guiding force during the generation process, ELSM achieves this by the use of initial community centers $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_K$, the initial membership vector $\mathbf{c}$ and the splitting mechanism as explained in §2.3.

However, during inference, given a sequence of observed adjacency matrices $\{\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, ..., \mathbf{A}^{(T)}\}$ for a dynamic network, the required guiding force is provided by the need to explain the observed data. In this case one need not incorporate any of the elements mentioned above. Thus for the purpose of inference, we eliminate these elements and get a simplified model which we call iELSM. The graphical model for iELSM is shown in Figure 2. Note that iELSM also captures both the motivating phenomena (similar nodes connect more often and nodes become similar to their neighbors over time) mentioned in the beginning of §2.

The structures of probability distributions that govern iELSM are similar to the structures of corresponding distributions in ELSM. The latent embeddings $\mathbf{z}_i^{(t)}$ follow (5) and (6) for $i = 1, 2, ..., n$ and $t = 2, 3, ..., T$. The observed entries $a_{ij}^{(t)}$ follow (2). In addition we impose a prior on the first layer of latent embeddings $\mathbf{z}_i^{(1)}$ for $i = 1, 2, ..., n$ as:

$$\mathbf{z}_i^{(1)} \sim \mathcal{N}(\mathbf{m}, s^2\mathbf{I}), \tag{8}$$

Where $\mathbf{m}$ and $s$ are hyperparameters. We make the following independence assumptions:

1. The latent embeddings in the first layer $\mathbf{z}_i^{(1)}$ for $i = 1, 2, ..., n$ are independent.

2. $a_{ij}^{(t)}$ is independent of everything else given $\mathbf{z}_i^{(t)}$ and $\mathbf{z}_j^{(t)}$ for all $i, j = 1, 2, ..., n, i \neq j$ and $t = 1, 2, ..., T$.

3. $\mathbf{z}_i^{(t)}$ is independent of everything else given $\mathbf{Z}^{(t-1)}$ and $\mathbf{A}^{(t-1)}$ for $i = 1, 2, ..., n$ and $t = 2, 3, ..., T$

Since the observed adjacency matrices are assumed to be symmetric and self-loops are not allowed, we only need to consider the lower triangle (without the leading diagonal) in these matrices. The joint log likelihood for iELSM, which is given below, can then be computed using (2), (5), (6) and (8) as:

$$\log P(\mathbf{A}^{(1)}, \mathbf{Z}^{(1)}, \mathbf{A}^{(2)}, \mathbf{Z}^{(2)}, ..., \mathbf{A}^{(T)}, \mathbf{Z}^{(T)}) =$$
$$\sum_{i=1}^{n} \log P(\mathbf{z}_i^{(1)}) + \sum_{t=1}^{T}\sum_{i>j} \log P(a_{ij}^{(t)}|\mathbf{z}_i^{(t)}, \mathbf{z}_j^{(t)})$$
$$+ \sum_{t=2}^{T}\sum_{i=1}^{n} \log P(\mathbf{z}_i^{(t)}|\mathbf{Z}^{(t-1)}, \mathbf{A}^{(t-1)}) \tag{9}$$

## 3.2 Variational Inference for iELSM

Exact inference is intractable in iELSM because the observed entries of $\mathbf{A}^{(t)}$ do not follow a Gaussian distribution. Thus, we resort to variational techniques (Blei, Kucukelbir, and McAuliffe 2017) to perform approximate inference. We use a LSTM based neural network to model the variational parameters. The neural network is then trained to maximize the ELBO which we derive next.

We approximate the true posterior distribution $P(\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)}, ..., \mathbf{Z}^{(T)}|\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, ..., \mathbf{A}^{(T)})$ using a distribution $Q(\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)}, ..., \mathbf{Z}^{(T)}; \theta)$, where $\theta$ represents parameters of a neural network. We further assume that $Q(., \theta)$ belongs to the mean-field variational family of distributions, i.e.:

$$Q(\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)}, ..., \mathbf{Z}^{(T)}; \theta) = \prod_{t=1}^{T}\prod_{i=1}^{n} q_{ti}(\mathbf{z}_i^{(t)}; \theta) \tag{10}$$

We model each factor $q_{ti}(\mathbf{z}_i^{(t)}; \theta)$ as a Gaussian distribution whose mean $\boldsymbol{\nu}_i^{(t)} \in \mathbb{R}^d$ and variance $(\boldsymbol{\sigma}_i^{(t)})^2 \in \mathbb{R}^d$ parameters are derived from the neural network. Here, as before, $d$ is the dimension of latent space. Note that we have not explicitly shown the dependence of $\boldsymbol{\nu}_i^{(t)}$ and $(\boldsymbol{\sigma}_i^{(t)})^2$ on $\theta$ to keep the notation clean. Let $(\boldsymbol{\sigma}_i^{(t)})^2\mathbf{I} = diag((\sigma_i^{(t)})_1^2, (\sigma_i^{(t)})_2^2, ..., (\sigma_i^{(t)})_d^2)$, we can then write:

$$q_{ti}(\mathbf{z}_i^{(t)}; \theta) = \mathcal{N}(\mathbf{z}_i^{(t)} \mid \boldsymbol{\nu}_i^{(t)}, (\boldsymbol{\sigma}_i^{(t)})^2\mathbf{I}) \tag{11}$$

The objective is to maximize the ELBO function which is given by $\mathbb{E}_{\mathbf{z}\sim q}[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z})]$ (Blei, Kucukelbir, and McAuliffe 2017). Here $\mathbf{x}$ denotes the observed variables, $\mathbf{z}$ denotes the latent variables, $p(\mathbf{x}, \mathbf{z})$ is the joint distribution of $\mathbf{x}$ and $\mathbf{z}$ and $q(\mathbf{z})$ is the distribution that approximates the posterior $p(\mathbf{z}|\mathbf{x})$. It can be shown that ELBO provides a lower bound on the log probability of observed data $p(\mathbf{x})$ (Kingma and Welling 2013), hence ELBO can be used as a surrogate function to maximize the probability of observed data. In our context, ELBO can be written as:

$$\text{ELBO}(\theta) = \mathbb{E}_{\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)}, ..., \mathbf{Z}^{(T)}\sim Q}[$$
$$\log P(\mathbf{A}^{(1)}, \mathbf{Z}^{(1)}, ..., \mathbf{A}^{(T)}, \mathbf{Z}^{(T)}) \tag{12}$$
$$- \log Q(\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)}, ..., \mathbf{Z}^{(T)}; \theta)]$$

The expectation of second term in (12) can be computed in closed form since $Q(., \theta)$ factorizes according to (10) and the factors $q_{ti}(.; \theta)$ follow a Gaussian distribution. To compute the expectation of first term, we use the Monte Carlo method to evaluate (9). In all our experiments, only one sample was used to approximate the expectation of first term.

## 3.3 Network Architecture

Given the variational parameters $\boldsymbol{\nu}_i^{(t)}$ and $(\boldsymbol{\sigma}_i^{(t)})^2$ for all $i = 1, 2, ..., n$ and $t = 1, 2, ..., T$, ELBO can be approximated using (12) as described above. In this section we describe the neural network architecture that parameterizes these quantities. There are two main advantages of this neural network based approach: (i) neural networks are powerful function approximates and hence complex node to latent embedding mappings can be learned and (ii) standard backpropagation algorithm with the reparameterization trick (Kingma and Welling 2013) can be used that allows efficient implementation.

The recurrent neural network is a natural model for sequential data. We use a bidirectional LSTM in our inference network. At time step $t$, our network takes the adjacency matrix $\mathbf{A}^{(t)}$ as input. Each row is treated as an input feature
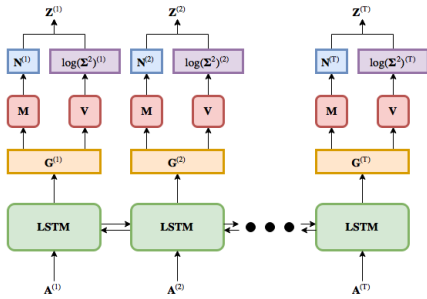
Figure 3: Inference network architecture for iELSM. $\mathbf{M}$ and $\mathbf{V}$ are neural networks that take $\mathbf{g}_i^{(t)}$ as input to produce the mean and log-variance vectors respectively. Note that these networks are shared across all timesteps.

vector. The network updates its hidden state using the usual update rules for LSTM and produces an output $\mathbf{g}_i^{(t)} \in \mathbb{R}^m$ for each node. The output $\mathbf{g}_i^{(t)}$ is then used to produce $\boldsymbol{\nu}_i^{(t)}$ and $\log (\boldsymbol{\sigma}_i^{(t)})^2$ via the mean and variance networks ($\mathbf{M}$ and $\mathbf{V}$ respectively in Figure 3) that are shared across all time steps. Figure 3 shows the architecture of our inference network.

We use the reparameterization trick for Gaussian distribution, as it was used in (Kingma and Welling 2013) to make the whole pipeline differentiable. This enables end to end training of the network. The embeddings sampled from this network can be used to compute (12) via Monte Carlo approximation as it was discussed in §3.2. The objective is to maximize $\mathrm{ELBO}(\theta)$ with respect to the parameters $\theta$ of the inference network.

### 3.4 Encoder-Decoder View of the Inference Network

The network given in Figure 3 can be seen as an encoder. At time step $t$, it takes the feature vector for each node (the row corresponding to that node in $\mathbf{A}^{(t)}$) as input and produces a time dependent embedding $\mathbf{z}_i^{(t)}$ for the node as output. Note that if observed node features are available, one can use these features in conjunction with $\mathbf{A}^{(t)}$ as input to the network. A decoder should take these latent embeddings and try to reconstruct $\mathbf{A}^{(t)}$. By using (9) in (12), we obtain a term that involves $\log P(a_{ij}^{(t)}|\mathbf{z}_i^{(t)}, \mathbf{z}_j^{(t)})$. The estimator of this probability value acts as a simple decoder that tries to predict the probability of an observed edge given the latent embeddings of its two endpoints.

Recall from (2) that: (i) $a_{ij}^{(t)}$ has been modeled by a Bernoulli distribution and (ii) The parameter for this Bernoulli distribution is given by $f(\mathbf{z}_i^{(t)} - \mathbf{z}_j^{(t)})$ where for generation we use $f(x) = 1 - \tanh(||x||_2^2/s_2^2)$. One can have a more complex decoder which utilizes problem specific information. For example, the function $f(x)$ can be modeled using a neural network. If the weights do not follow the Bernoulli distribution (i.e., the edges are weighted)

then one can use the problem context to enforce the right distribution on weights and learn the parameters of that distribution accordingly. For example, in our experiments with Enron dataset, the edge weights are non-negative integers, so we model this by using a Poisson distribution and we use a neural network to learn $f(x)$ so that it predicts the mean for the Poisson distribution.

This encoder-decoder view of the inference network makes the inference methodology very flexible. One can incorporate domain knowledge to customize various components of this network so that it better suits the problem at hand.

## 4  Related Work

In general, dynamic networks, and in particular, link prediction and community detection in dynamic networks, have earned the attention of many researchers (Goldenberg et al. 2010; Kim et al. 2017). This can be attributed to the emergence of many practical applications (for e.g., social network analysis) where dynamic networks are encountered.

An extension of Mixed Membership Stochastic Block Model (Airoldi et al. 2008) to the dynamic network setting by coupling it with a state-space model to capture temporal dynamics has been proposed in (Xing, Fu, and Song 2010; Ho, Song, and Xing 2011). Along the same lines, in (Yang et al. 2011), Stochastic Block Model (Holland, Laskey, and Leinhardt 1983) was extended by explicitly modeling transition between communities over time. Some other approaches that extend a static network models are (Xu and Hero 2014; Xu 2015; Papadopoulos et al. 2012).

There have been many latent space based approaches for modeling dynamic networks. In (Sewell and Chen 2015; 2016) the transition of nodes' latent embeddings is modeled independently for each node by using a Markov chain. In (Heaukulani and Ghahramani 2013), the authors introduced Latent Feature Propagation (LFP) that uses a discrete latent space and a HMM style latent embedding transition model. In (Kim and Leskovec 2013) non-parametric Dynamic Multigroup Membership Graph model (DMMG) which additionally models the birth and death of communities using Indian Buffet Process (Griffiths and Ghahramani 2011) has been introduced. The work of (Miller, Jordan, and Griffiths 2009; Foulds et al. 2011) are also along the same lines.

Most similar to our work, is the work done in (Heaukulani and Ghahramani 2013; Kim and Leskovec 2013), however there are significant differences as well. (i) While these approaches rely on MCMC, we use variational inference because it offers many advantages as mentioned in (Blei, Kucukelbir, and McAuliffe 2017), (ii) We use a neural network to perform inference which allows an efficient implementation and (iii) We have a more flexible, continuous latent space whereas these approaches use a discrete latent space.

## 5  Experiments

In this section we demonstrate the applicability of our model on two tasks that have been widely studied - community detection and link prediction. We perform experiments on syn-

thetic networks generated from ELSM and real world networks. Our model outperforms existing methods on standard benchmarks. While we report the results for both ELSM and iELSM in §5.3 and §5.4, the inference procedure for ELSM has been described in supplementary material.

## 5.1 Synthetic Networks

We generated 10 synthetic networks using Algorithm 1. For each network, we used $n = 100$, $T = 10$, $K = 5$, $\pi = [1/K, ..., 1/K]$, $\mathbf{m} = [0, 0]^\mathsf{T}$, $s = 1.0$, $s_1 = 0.05$, $s_2 = 0.2$, $s_3 = 1.0$ and $s_4 = 0.5$ as input parameters. Community detection was performed on these networks using both normalized spectral clustering (von Luxburg 2007) and our inference network as described in §5.3. The scores reported in Table 1 for synthetic networks were obtained by averaging the scores across these 10 networks. Visualization of some generated networks and the corresponding latent spaces has been provided in supplementary material. A video that shows the evolution of latent space over time has also been attached as part of supplementary material.

## 5.2 Real World Networks

We evaluate our model on three real world dynamic networks apart from evaluating it on synthetic networks generated by ELSM. This section describes the datasets that we have used for our experiments:

**1) Enron email:** The Enron dataset (Klimt and Yang 2004) contains emails that were exchanged between 149 individuals over a period of three years. We use two different versions of this dataset - Enron-full and Enron-50. Enron-full has 12 snapshots, one for each month of the year 2002. In snapshot $t$, entry $a_{ij}^{(t)}$ counts the number of emails that were exchanged between users $i$ and $j$ during that month. We also consider unweighted Enron-full obtained by making all snapshots binary. Following (Foulds et al. 2011), Enron-50 considers only top 50 individuals with most number of emails across all snapshots. Each snapshot corresponds to a month (there are 37 snapshots). The adjacency matrix for each snapshot is symmetric and binary where $a_{ij}^{(t)} = 1$ if at least one email was exchanged between users $i$ and $j$ during that month and 0 otherwise.

**2) NIPS co-authorship:** There are 17 snapshots in this network, each corresponding to an year from 1987 to 2003. Entry $a_{ij}^{(t)}$ counts the number of NIPS conference publications in year $t$, that have individuals $i$ and $j$ as co-authors. We use a subset of this dataset - NIPS-110, that is created by making the snapshots binary and selecting top 110 authors based on the number of unique co-authors that they have had over these 17 years (Heaukulani and Ghahramani 2013).

**3) Infocom:** This dataset contains information about physical proximity between 78 individuals over a 93 hours long interval at Infocom 2006. Following (Kim and Leskovec 2013), we create snapshots corresponding to 1 hour long intervals. At time $t$, $a_{ij}^{(t)} = 1$ if both individuals $i$ and $j$ registered each others physical proximity during the $t^{th}$ hour. We remove those snapshots that have fewer than 72 non-zero entries which leaves us with 50 snapshots.

We next describe the two tasks that we have performed on various subsets of these datasets, namely - community detection and link prediction.

## 5.3 Community Detection

The task is to assign each node to a community at each time step such that *similar* nodes belong to the same community. To measure the quality of communities, at each time step, we use the well known modularity score. The modularity score lies in the range $[-1/2, 1)$, with values close to 1 signifying *good* communities (that have more edges within them as compared to the number of edges that will be obtained purely by chance).

We use the observed matrices $\{\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, ..., \mathbf{A}^{(T)}\}$ to train the inference network for iELSM (§3.3) and ELSM (supplementary material). The latent embeddings obtained from the trained network are fed to the k-means clustering algorithm. At each time step, the optimal number of communities, $k^{(t)}$ is chosen from the range $[2, 10]$ by selecting the value of $k^{(t)}$ that maximizes the modularity score on the adjacency matrix that is induced by applying RBF kernel with variance parameter 1 to the latent space embeddings. Note that we do not use the original adjacency matrix $\mathbf{A}^{(t)}$ while selecting the optimal number of communities.

We experiment with different variants of decoder in the inference network. For Enron-full, the edges have positive integer weights, we model this by imposing a Poisson distribution on $P(a_{ij}^{(t)}|\mathbf{z}_i^{(t)}, \mathbf{z}_j^{(t)})$. Since iELSM (and ELSM) models the interaction between nodes as a function of distance between their latent embeddings, we predict the mean of Poisson distribution for position $(i, j)$ at time step $t$ as:

$$\rho_{ij}^{(t)} = \exp(-w_\rho^2 ||\mathbf{z}_i^{(t)} - \mathbf{z}_j^{(t)}||^2 + b_\rho) \qquad (13)$$

The parameters $w_\rho$ and $b_\rho$ are shared across all time steps and positions. These parameters are learned using standard backpropagation ((13) represents a single layer, single node neural network with $\exp(.)$ as the activation function). One can also employ a similar technique for learning parameters $s_2$ (§2.2) and $s_4$ (§2.4) from data by using a single layer, single node neural network to learn optimal scaling of distances in (2) and (5).

The key claim that we wish to demonstrate here is that the latent embeddings learned by our model are not only good for finding communities at individual time steps, but also lead to a more plausible, gradually changing community structure over time. To do so, we compute the NMI score between community assignments at successive time steps. The NMI score lies in $[0, 1]$ with values close to 1 signifying gradual change in community structure in our setup.

We compare our scores against the scores obtained by independently applying normalized spectral clustering (von Luxburg 2007) at each snapshot. The number of communities, $k^{(t)}$, is chosen by using the same method that was used for our model on the node embeddings produced by spectral clustering.

Table 1 summarizes the average modularity and NMI scores for various datasets. Although $k^{(t)}$ is chosen based

Table 1: Community Detection Results: It can be seen that the detected communities are meaningful (as evident by modularity scores that are comparable to the ones obtained by the spectral clustering algorithm) while at the same time being *smooth* (since the NMI scores are considerably higher) thereby validating our claim.

| DATASET | MODULARITY | | | NMI | | |
|---|---|---|---|---|---|---|
| | SPECTRAL | iELSM (OURS) | ELSM (OURS) | SPECTRAL | iELSM (OURS) | ELSM (OURS) |
| SYNTHETIC | .479 | **.489** | .488 | .769 | .851 | **.864** |
| ENRON-FULL (WEIGHTED) | .506 | **.597** | .590 | .455 | .722 | **.823** |
| ENRON-FULL (BINARY) | .540 | **.555** | .551 | .529 | .767 | **.779** |
| ENRON-50 | .396 | **.419** | .414 | .560 | .819 | **.838** |
| NIPS-110 | .497 | **.601** | .595 | .249 | .804 | **.863** |
| INFOCOM | .283 | **.288** | .270 | .443 | .643 | **.662** |

Table 2: Link Prediction Results: Our method outperforms other approaches on both metrics. We were unable to obtain an implementation for DMMG and hence the performance numbers of DMMG on Enron-50 are missing.

| | ENRON-50 | | INFOCOM | | NIPS-110 | |
|---|---|---|---|---|---|---|
| | AUC | F1 | AUC | F1 | AUC | F1 |
| BAS | .874 | .585 | .698 | .317 | .703 | .161 |
| LFRM | .777 | .312 | .640 | .248 | .398 | .011 |
| DRIFT | .910 | .578 | .782 | .381 | .672 | .084 |
| DMMG | - | - | .804 | .392 | .732 | .196 |
| iELSM (OURS) | **.913** | **.600** | .868 | **.489** | **.754** | .248 |
| ELSM (OURS) | .911 | .596 | **.871** | **.489** | .742 | **.251** |

on the graph induced by latent embeddings, the scores reported in Table 1 correspond to the observed graph for the chosen number of communities. From Table 1, it is clear that our model achieves modularity scores that are at par (or better) with respect to spectral clustering while exhibiting a significantly higher NMI score, thus validating our claim.

## 5.4 Link Prediction

Good performance on community detection task testifies that the latent embeddings being learned by the network have a nice structure. But how does one ensure that the modeled network dynamics are faithful to real world data? Is our model overfitting to the training data to get good embeddings? These questions get answered, if the observed data can be extrapolated to predict an unseen network snapshot using our model. This task is known as link prediction.

Formally, given network snapshots up to time $t$, $\{\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, ..., \mathbf{A}^{(t)}\}$, we want to predict the next network snapshot $\mathbf{A}^{(t+1)}$. Note that this involves predicting the appearance of new links as well as removal of existing links. We use binary networks in this experiment to compare against other approaches. We use the well known AUC score and F1 score for the purpose of comparison. Values close to 1 indicate good performance for both the scores.

To predict $\mathbf{A}^{(t+1)}$, we train the inference network for iELSM and ELSM on $\{\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, ..., \mathbf{A}^{(t)}\}$. We update the predicted latent embeddings $\mathbf{Z}^{(t)}$ using (5) and (6) to get $\mathbf{Z}^{(t+1)}$. Then, $P(a_{ij}^{(t+1)} = 1 | \mathbf{Z}^{(t+1)})$ is computed using (2) (or the decoder, if a decoder network has been used). Note that for ELSM, while updating the latent embeddings we set $h_i^{(t+1)} = 0$ for all nodes.

We compare our performance against a simple baseline (BAS) in which the probability of an edge is directly proportional to the number of times it has been observed in the past (Foulds et al. 2011). We also compare against existing approaches - LFRM (Miller, Jordan, and Griffiths 2009) (using only the last snapshot), DRIFT (Foulds et al. 2011) and DMMG (Kim and Leskovec 2013) (Table 2). Maximum F1 score over all thresholds is selected at each snapshot as it was done in (Kim and Leskovec 2013). The scores reported here have been obtained by averaging the snapshot wise scores. It can be seen that our method outperforms other methods on both metrics. Visualization of latent embeddings and predicted output matrices for Enron-full can be found in supplementary material.

## 6 Conclusion and Future Work

In this paper, we proposed ELSM, a generative model for dynamically evolving networks. We also proposed a neural network architecture that performs approximate inference in a simplified version of our model, iELSM (inference for ELSM is in supplementary material) and highlighted the flexibility of this approach. Our model is capable of: (i) Generating synthetic dynamic networks with gradually evolving communities and (ii) Learning meaningful latent embeddings of nodes in a dynamic network. We demonstrated the quality of learned latent embeddings on downstream tasks like community detection and link prediction in dynamic networks.

In this paper we focused on undirected, positive influence based, gradually changing, assortative networks with a fixed number of nodes. Though, these properties are exhibited by

a large number of real world networks, there are other important classes of networks that do not follow these properties. For example, one can also extend this idea to directed networks. One can also consider the case where the number of nodes is allowed to change over time. Considering networks that are not necessarily assortative (like hierarchical networks) also poses interesting questions.

# References

Airoldi, E. M.; Blei, D. M.; Fienberg, S. E.; and Xing, E. P. 2008. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research (JMLR)* 9:1981–2014.

Blei, D. M.; Kucukelbir, A.; and McAuliffe, J. D. 2017. Variational inference: A review for statisticians. *Journal of the American Statistical Association* 112(518):859–877.

Fortunato, S. 2010. Community detection in graphs. *Physics Reports* 486(3):75 – 174.

Foulds, J.; DuBois, C.; Asuncion, A.; Butts, C.; and Smyth, P. 2011. A dynamic relational infinite feature model for longitudinal social networks. *Proceedings of Machine Learning Research (PMLR)* 15:287–295.

Goldenberg, A.; Zheng, A. X.; Fienberg, S. E.; and Airoldi, E. M. 2010. A survey of statistical network models. *Foundations and Trends®in Machine Learning* 2(2):129–233.

Griffiths, T. L., and Ghahramani, Z. 2011. The indian buffet process: An introduction and review. *Journal of Machine Learning Research (JMLR)* 12:1185–1224.

Gupta, S.; Sharma, G.; and Dukkipati, A. 2018. A generative model for dynamic networks with applications: Supplementary material. *ArXiv e-prints* 1802.03725.

Heaukulani, C., and Ghahramani, Z. 2013. Dynamic probabilistic models for latent feature propagation in social networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*.

Ho, Q.; Song, L.; and Xing, E. P. 2011. Evolving cluster mixed-membership blockmodel for time-varying networks. *Journal of Machine Learning Research (JMLR)* 15:342–350.

Holland, P. W.; Laskey, K. B.; and Leinhardt, S. 1983. Stochastic blockmodels: First steps. *Social Networks* 5(2):109 – 137.

Kim, M., and Leskovec, J. 2013. Nonparametric multi-group membership model for dynamic networks. *Advances in Neural Information Processing Systems* 26:1385–1393.

Kim, B.; Lee, K.; Xue, L.; and Niu, X. 2017. A review of dynamic network models with latent variables. *CoRR*.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*.

Klimt, B., and Yang, Y. 2004. The enron corpus: A new dataset for email classification research. In *In Proceedings of the European Conference on Machine Learning (ECML)*, volume 3201.

Miller, K.; Jordan, M. I.; and Griffiths, T. L. 2009. Nonparametric latent feature models for link prediction. *Advances in Neural Information Processing Systems* 22:1276–1284.

Papadopoulos, F.; Kitsak, M.; Serrano, M. n.; Boguñá, M.; and Krioukov, D. 2012. Popularity versus similarity in growing networks. *Nature* 489.

Rossetti, G., and Cazabet, R. 2017. Community discovery in dynamic networks: A survey. *CoRR* abs/1707.03186.

Sewell, D. K., and Chen, Y. 2015. Latent space models for dynamic networks. *Journal of the American Statistical Association* 110(512):1646–1657.

Sewell, D. K., and Chen, Y. 2016. Latent space models for dynamic networks with weighted edges. *Social Networks* 44:105 – 116.

von Luxburg, U. 2007. A tutorial on spectral clustering. *Statistics and Computing* 17(4):395–416.

Xing, E. P.; Fu, W.; and Song, L. 2010. A state-space mixed membership blockmodel for dynamic network tomography. *The Annals of Applied Statistics* 4(2):535–566.

Xu, K. S., and Hero, A. O. 2014. Dynamic stochastic block-models for time-evolving social networks. *IEEE Journal of Selected Topics in Signal Processing* 8(4):552–562.

Xu, K. S. 2015. Stochastic block transition models for dynamic networks. *Journal of Machine Learning Research (JMLR)* 38:1079–1087.

Yang, T.; Chi, Y.; Zhu, S.; Gong, Y.; and Jin, R. 2011. Detecting communities and their evolutions in dynamic social networks - a Bayesian approach. *Machine Learning* 82(2):157–189.