Network Change Detection Based on Random Walk in Latent Space

Chuan-Hao Lin

January, 2021

1 Motivation

The research addresses change detection of the network structures over time, and develops an algorithm for the problem. A change is often considered as a significant deviation from the previous data. However, network structures are too complicated to analyze the difference directly. In the proposed algorithm, the concept of the latent space is applied, where the nodes in networks are relocated via network embedding, and then random walk process is performed to explore the networks. It is known that transforming nodes into the latent space is an effective way to present the networks in relatively low-dimensional space. It also helps decrease the noise of adjacency matrices. The positions of the nodes in the latent space reveal the relationships of nodes. The closer the nodes are with each other, the higher chance they are connected with edges in the networks. The edges probabilities can thus be constructed via the relative positions of nodes, while there are no actual edge representations in the latent space. Then, random walk is adopted to export a descriptor for the network, and the network change detection problem can be transformed to the change detection over the descriptors. The experiments show that the algorithm can detect the changes effectively, with faster reactions and less false alarms, comparing with other methods that do not employ the latent space. The accuracy and stability of the algorithm point out the importance of positioning nodes into the latent space, where a general criterion to describe the relationships of nodes and edges is provided.

2 Network Embedding

The embedding method maps a node to a vector, displaying its position in latent space, i.e. $Embedding : V \to \mathbb{R}^d$, where d is the dimensionality of the latent space.

With the transformation, no explicit connections between nodes are presented anymore. Instead, all the nodes are located in the latent space, in which the distance shows their relationship between each others. Closer distance implies stronger relationship, and vice versa. Therefore, a clear, general criterion emerges, and the relationship can be depicted via this feature.

On the other hand, a recovering function thrives to resume the relationship between nodes. That is, given a set of nodes in the latent space, it tries to restore the relationship between all the provided nodes. The function performs like $Recovery : \mathbb{R}^{d\times m} \to \mathbb{R}^+$, where m is the amount of input nodes. While fully recovery is not guaranteed, it is the goal of network embedding to do its best to reduce the loss of prediction.

One of the recovering idea is pairwise recovery, which focuses on only two nodes at a time. The recovering function $Pairwise\ Recovery : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ has the output to be a recovered weight.

3 Transition Probability and Stationary Vector

Note that while the graphs from the time series pass one by one, the transformation process does not necessarily treat all the graphs in the similar fashion. In another word, the same node may endow completely different position throughout each timestamps even though the original graphs are similar. Therefore, it is meaningless to consider the absolute position of nodes in the latent space.

Instead, the relative positions are much more meaningful, since the relative distance show the relationship between nodes, and the strength should hold still under time series if no significant changes take place. To interpret the relationship in practice, we apply the random walk model, which gives a set of transition probability to each pair of nodes. In another word, the model provides some sense of the probability under which the edges between the given nodes will appear. In this sense, we can build a transition matrix for each timestamp.

Let the transition matrix be $P(t)$, where t denotes as the timestamp. There exists some unique equilibrium point in the random walk model as long as there is only one eigenvalue equal to 1. The corresponding eigenvector will be the vector characterizing the whole graph. The vectors are called stationary vectors.

The stationary vectors represent the stable states in the random walk. Those elements represent the probability on the certain node. Therefore, all the elements should lie within $[0, 1]$, and their norm should be normalized.

4 Probability Estimation

So far, we start from the transition matrices, standing for the probability of the edges. Then the nodes align correspondingly, which are the stationary vectors. The whole process is determined simultaneously. Thus, one can actually view the generation of graphs conversely. That is, the nodes come out with the probability, and the edges follow under the scenario.

In such viewpoint, one can treat the generation of the nodes as a multinomial process, with the stationary vector serving as the underlying probability for each node throughout the timestamps. To explore a suitable probability distribution for the multinomial process, one of the candidates is the conjugate distribution of multinomial distribution, Dirichlet distribution, which serves the same forms of prior probability and posterior probability in multinomial process.

Dirichlet distribution has the following form

$$Dir(\boldsymbol{\mu}|\boldsymbol{\alpha}) = \Gamma(\sum_{i=1}^{N} \alpha_i) \prod_{i=1}^{N} \frac{\mu_i^{\alpha_i - 1}}{\Gamma(\alpha_i)}, \tag{1}$$

which endows a parameter vector α of N-dimension, and μ_i stands for the element of the stationary vector. Then, a change indicates the parameter vector has shifted.

Note that Dirichlet distribution satisfies the two requirements of stationary vectors. That is, all the elements range from $[0, 1]$, and the 1-norm is normalized to 1.

There is no close form for the derivatives of Dirichlet distribution. To estimate the parameters, one can derive the maximum likelihood estimation with Newton method [Estimating a Dirichlet distribution, Minka, 2000]. However, to estimate the distribution, one require a abundant of data to have a reasonable, stable result. Thus, we employ a window with size w, in which the data are used to estimated one probability distribution.

5 Change Detection

There are a lot of existing methods that allows us to define the change once we represent the networks as vectors. The straight-forward way is to measure the difference between probability distributions, such as KL-divergence or MDL-change statistics. Further, a method that aggregates different model can be applied if the result of model selection is unclear. In addition, to detect the changes in real-time, dynamic threshold and adaptive windowing is also available.