1 Li, Dani, Hu, Tang, Chang, and Liu: Attributed Network Embedding for Learning in a Dynamic Environment

1.1 Introduction

Considering similarities between nodal attributes when performing network embedding can often lead to more accurate community detection performance than just considering the information contained in a standard normalzied or unnormalized graph Laplacian. Especially in networks with just a few nodes or edges, traditional algorithms for spectral clustering and the like that do not consider nodal attributes perform poorer and poorer at community detection. In considering the nodal attributes, the clustering process becomes easier. This is in part due to *homophily*, a social science term that describes how we are more likely to form connections with people with more than just one similar attribute to us.

Existing literature also does not account for the fact that real networks tend to change over time, with shifting attributes and added/deleted nodes and edges. It uses what the paper defines to be a ”static” network, with a graph Laplacian and node attribute vector that does not change over time. On the contrary, this paper focuses on **dynamic attributed networks**, networks that exhibit network and node attribute value changes over time.

Some difficulties one might encounter in trying to analyze these dynamic attributed networks:

1. Network attributes and nodal attributes are often correlated, so noise in the description of either one of these attributes will compound tremendously.

2. Re-embedding network data with each update to the network or nodal attribute data is computationally intensive.

The algorithm proposed by this paper creates an offline embedding at initial time t, which is then updated by some online embedding for each subsequent time period. Switching to an online model after an initial offline model addresses many of the issues in the second difficulty in dynamic attributed network analysis described above, as it is much less computationally intensive.

For notations sake, let A be the adjacency matrix of a network and $X^{(t)}$ be a $n \times d$ matrix of nodal attributes such that ΔA represents the change in the adjacency matrix of a network across periods and ΔX represents the change in a network's nodal attributes.

1.2 Proposed Framework

1.2.1 Initial Offline Model

To simultaneously account for both network and nodal attributes, an embedding that is a consensus between these two streams of information is necessary.

Because of the aforementioned first difficulty in analyzing dynamic attributed networks, noise needs to be reduced or else it will be compounded when we combine information from both sources. The chosen way to reduce noise is to perform spectral embedding for both the network attribute matrix and nodal attribute matrix.

For the network attribute matrix, simple unnormalized spectral clustering is performed. Instead of further clustering the eigenvalue outputs in k-dimensional space using something like k-means, we simply stop the clustering procedure and denote the first k eigenvectors after the first to be the vector $Y_A^{(t)}$

For the nodal attribute matrix, a similarity matrix W^t is constructed using the cosine similarity matrix. This similarity matrix is then taken to be the equivalent of a graph Laplacian, and for this matrix W^t we denote the first k eigenvectors after the first to be the vector $Y_X^{(t)}$.

One then generates some Y^t such that this vector of eigenvectors is the result of an optimization problem in which the correlation between $Y_A^{(t)}$ and $Y_X^{(t)}$ is maximized.

1.2.2 Subsequent Online Model

In the online model, the objective is to simplify updating the network and nodal attribute matrices such that one need not compute all the eigenvectors of each matrix each time either attribute matrix is updated. In other words, one would like to simplify the calculation of a given eigenvector i of the graph Laplacian $A^{(t)} + \Delta A$ without having to calculate all the eigenvectors of $A^{(t)} + \Delta A$.

Updating eigenpair (λ_i, a_i) can be obtained through the following equations.

Once one has updated both $Y_A^{(t)}$ and $Y_X^{(t)}$, one reperforms the optimization problem to get a new Y^t.

This procedure is shown to be more computationally efficient and accurate than any predecessor.