Hierarchical Latent Space Models
for Multiplex Social Network Analysis

Alex Martin Loewi
aloewi@cmu.edu

Francisco Ralston Fonseca
fralston@andrew.cmu.edu

Octavio Mesner
omesner@cmu.edu

Abstract

abstract...

1 Introduction

Testing citations [**?**], [**? ?**], [**? ? ?**].

1.1 The Curious Case of SNA

Social network analysis (SNA) is a complex, and inherently interdisciplinary field. It takes the already complex fields of psychology and sociology, and focuses on the most interdependent processes in these disciplines. This then creates systems so complicated that sophisticated statistical models are necessary to make any sense of them. While this complexity makes SNA an exciting intersection, effectively spanning these two disparate approaches is exceptionally difficult. As a result, while it attracts a great deal of attention, many important problems in SNA are under-studied, simply because of the small number of people who are trained to deal with both the substance, and methods, of the field.

One of these problems is that of modeling multigraphs. Multigraphs, also called multiview networks, can be thought of as either a single graph with multiple edge types, or as multiple networks over the same set of nodes. Data of this kind is not uncommon, at either large or small scales. Twitter has Following relationships, as well as Retweets, and Likes, each of which can be thought of as simply one layer within the full relationship between two nodes. In small scales, surveys for networks very rarely ask about a single type of relationship, and many of the most famous network data sets have multiple relationships (such as Trust, Friendship, and Respect, in Samson's monk data). Despite the existence of data of this form for decades, the strange nature and inherent difficulty of this data has led to an extremely small number of ways to model it, all of which have serious shortcomings. The models from statistics tend to ignore the problems of the practitioner, and the practitioners rarely have the training to develop their own methods.

1.2 The Practical Difficulties of Multigraphs

In particular, practitioners have two immediate problems when dealing with multigraphs, both of which are consequences of the inherent complexity of the data.

1.2.1 Visualization

The first problem is that multigraphs are too complex to easily visualize, and visualization has always been a cornerstone of network analysis. In particular, the difficulty comes in comparing the many layers of data (i.e. the graphs formed by the different edge types). Because two layers have the same set of nodes, comparing them requires the nodes to be in similar positions for the two layouts. However, graph layout algorithms are designed to pay attention only to the connectivity in a graph,

and find the node positions that best reflect that connectivity – if the node positions are fixed, all of the meaningful structure in a layer may be obscured. To define the problems in terms of its extremes, there is a trade-off in terms of comparability (fixed nodes) and expressiveness (free nodes). A useful multigraph model would be able to tune explicitly between these two extremes.

1.2.2 Dimensionality

The second problem is that multigraphs may be unnecessarily complex. Empirically, there are often high degrees of correlation between the layers in a multigraph, and the smaller the number of graphs, the easier analysis becomes. These two observations combine to suggest another useful property of a new model – the ability to remove redundant layers. While existing models often take the data and make it even more complicated, that runs counter to the needs of an applied social network analyst.

2 The Model

2.1 Problem Statement

Motivated by these problems, we propose a model with the following three objectives:

1. Use current methods from statistical SNA to model multigraphs

2. Model the layers of the multigraph in a way that can tune for comparability

3. Allow the model to remove redundant layers

A model with all of these qualities, which we term the Hierarchical Latent Space Model, can be described in the following way:

2.2 The Likelihood Function

Using the canonical Latent Space Model [?] as a starting point, we model a set of conditionally independent binary dyads.

The existence of each of these dyads is a function of an intercept, a set of covariates (omitted here for clarity), and the distance between the latent "position" variables z of the two nodes in the dyad. The goal of the model will be to estimate these positions. An optimal set of variables would place positions close together for nodes with an observed edge, and vice versa.

$$\eta_{ijk} = \alpha_k - \|z_{ik} - z_{jk}\|_2^2$$

The indices i and j are over the nodes; the index k refers to the different layers in the multigraph. Because the edges are binary, we use the inverse logit σ to transform η, but it should be observed that for real-valued edges, other link functions could be easily substituted.

$$\sigma(\eta_{ijk}) = \frac{1}{1 + \exp(-\eta_{ijk})} \in [0, 1]$$

All together, the unpenalized likelihood of the HLSM thus takes the form of a binomial:

$$L = \prod_k \prod_i \prod_{j<i} \sigma(\eta_{ijk})^{y_{ijk}} (1 - \sigma(\eta_{ijk}))^{1-y_{ijk}}$$

and the log likelihood is

$$\ell = \sum_k \sum_i \sum_{j<i} y_{ijk} \ln \sigma(\eta_{ijk}) + (1 - y_{ijk}) \ln(1 - \sigma(\eta_{ijk}))$$

$$= \sum_k \sum_i \sum_{j<i} -y_{ijk} \ln(1 + \exp(-\eta_{ijk})) + (1 - y_{ijk}) \ln \left(\frac{\exp(-\eta_{ijk})}{1 + \exp(-\eta_{ijk})} \right)$$

$$= \sum_k \sum_i \sum_{j<i} -y_{ijk} \ln(1 + \exp(-\eta_{ijk})) + (1 - y_{ijk})[-\eta_{ijk} - \ln(1 + \exp(-\eta_{ijk}))]$$

$$= \sum_k \sum_i \sum_{j<i} -y_{ijk} \ln(1 + \exp(-\eta_{ijk})) - \eta_{ijk} - \ln(1 + \exp(-\eta_{ijk})) + y_{ijk}\eta_{ijk} + y_{ijk} \ln(1 + \exp(-\eta_{ijk}))]$$

$$= \sum_k \sum_i \sum_{j<i} (y_{ijk} - 1)\eta_{ijk} - \ln(1 + \exp(-\eta_{ijk}))$$

2.3 Regularization

While the notation z for the latent variables is consistent with the literature, and somewhat more intuitive, our contribution comes from a reparameterization of the model, where

$$z_{ik} = b_i + \epsilon_{ik}$$

This parameterization takes an explicit mid-point between the two most recently published papers on this problem. One [?] fits the layers independently, and allows correlations between them; another [] treats all of the layers simultaneously, as part of a single complex object. Our model allows not only for similar behaviors to both of these extremes, but additionally allows the user to tune between them manually.

Our model starts with a "base" position b_i for each node, which is the hierarchical layer in the model. It then adds a layer-specific perturbation ϵ_{ik}. The behavior of the model then depends entirely on the regularization placed on the ϵ's. When there is none, each layer is fit independently. When there is an arbitrarily large amount, the perturbations are all driven to zero, and all k layers are represented identically by the base parameters b. With intermediate values however, the user can find the point between these two extremes that allows for the graphs to be distinct, but also renders them sufficiently similar to be comparable, by not allowing the perturbations to stray too far from their shared base position.

Furthermore, with the use of a group lasso penalty applied to all of the perturbations within a single layer k, the model itself will perform graph-wise dimensionality reduction. This is far more valuable to the practitioner than an element-wise sparse solution, which would still require them to consider all of the many layers in their multigraph.

3 Fitting

3.1 The Optimization Problem

Together these two elements form our optimization problem, which is to minimize the sum of the negative log likelihood and a lasso penalty on the deviations ϵ_{ik}.

$$\min_{\epsilon, b} \sum_k \sum_i \sum_{j<i} [(1 - y_{ijk})\eta_{ijk} + \ln(1 + \exp(\eta_{ijk}))] + \lambda \sum_k \|\epsilon_k\|_2$$

where ϵ_k represents all of the perturbations for layer k. Since this objective is not differentiable, we have several options, but the clear first choice is to use proximal gradient descent.

3.2 Derivation of the Gradients

We define our decision variables as: $\{b_i\} \; \forall \, i$ and $\{\epsilon_{ik}\} \; \forall \, i, k$.

Then, the gradients of the η_{ijk} are:

$$\frac{\partial \eta_{ijk}}{\partial b_i} = -2(b_i + \epsilon_{ik} - b_j - \epsilon_{jk})$$

$$\frac{\partial \eta_{ijk}}{\partial b_j} = 2(b_i + \epsilon_{ik} - b_j - \epsilon_{jk})$$

$$\frac{\partial \eta_{ijk}}{\partial \epsilon_{ik}} = -2(b_i + \epsilon_{ik} - b_j - \epsilon_{jk})$$

$$\frac{\partial \eta_{ijk}}{\partial \epsilon_{jk}} = 2(b_i + \epsilon_{ik} - b_j - \epsilon_{jk})$$

$$\frac{\partial \ell}{\partial b_m} = \sum_k \sum_i \sum_{j<i} (y_{ijk} - 1) \frac{\partial \eta_{ijk}}{\partial b_m} - \frac{\partial \ln(1 + \exp(-\eta_{ijk}))}{\partial b_m}$$

$$= \sum_k \left[\sum_{j<m} (y_{mjk} - 1) \frac{\partial \eta_{mjk}}{\partial b_m} - \frac{\partial \ln(1 + \exp(-\eta_{mjk}))}{\partial b_m} + \sum_{i>m} (y_{imk} - 1) \frac{\partial \eta_{imk}}{\partial b_m} - \frac{\partial \ln(1 + \exp(-\eta_{imk}))}{\partial b_m} \right]$$

$$= \sum_k \left[-2 \sum_{j<m} \left[(y_{mjk} - 1)(b_m + \epsilon_{mk} - b_j + \epsilon_{jk}) + \frac{\exp(-\eta_{mjk})}{1 + \exp(-\eta_{mjk})} (b_m + \epsilon_{mk} - b_j - \epsilon_{jk}) \right] \right.$$

$$\left. + 2 \sum_{i>m} \left[(y_{imk} - 1)(b_i + \epsilon_{ik} - b_m - \epsilon_{mk}) + \frac{\exp(-\eta_{ijk})}{1 + \exp(-\eta_{mjk})} (b_i + \epsilon_{ik} - b_m - \epsilon_{mk}) \right] \right]$$

$$= 2 \sum_k \left[-\sum_{j<m} \left(y_{mjk} - 1 + \frac{\exp(-\eta_{mjk})}{1 + \exp(-\eta_{mjk})} \right) (z_{mk} - z_{jk}) \right.$$

$$\left. + \sum_{i>m} \left(y_{imk} - 1 + \frac{\exp(-\eta_{ijk})}{1 + \exp(-\eta_{mjk})} \right) (z_{ik} - z_{mk}) \right]$$

$$\frac{\partial \ell}{\partial \epsilon_{mk}} = 2 \left[-\sum_{j<m} \left(y_{mjk} - 1 + \frac{\exp(-\eta_{mjk})}{1 + \exp(-\eta_{mjk})} \right) (z_{mk} - z_{jk}) \right.$$

$$\left. + \sum_{i>m} \left(y_{imk} - 1 + \frac{\exp(-\eta_{ijk})}{1 + \exp(-\eta_{mjk})} \right) (z_{ik} - z_{mk}) \right]$$

3.3 The Proximal Operator

The proximal operator for the penalty term using the L2 norm[1] is:

$$\text{prox}_{\|\cdot\|,t}(\epsilon_{i,k}) = \begin{cases} \frac{\|\epsilon_{i,k}\| - \lambda t}{\|\epsilon_{i,k}\|} \epsilon_{i,k}, & \|\epsilon_{i,k}\| \geq \lambda t \\ 0, & \|\epsilon_{i,k}\| < \lambda t \end{cases}$$

$$\text{prox}_{\|\cdot\|,t}(b_i) = b_i$$

Therefore, let β be the vector with all parameters ϵ and b. The Proximal Gradient Descent method in this case is:

$$\beta^+ = \text{prox}_{\|\cdot\|,t}(\beta - t\nabla\ell(\beta))$$

---

[1] According to the results on Homework 3

4

3.4 Non-Convexity and Initialization

The original Latent Space Model is known to be non-convex in the latent positions z [?], and our model is no different. This issue is typically approached by using MCMC samplers [? ?] which are capable of exploring non-convex spaces, however these are known to have issues as well. Using them on this problem proved not only to take far more time, but also to have substantially worse results. We restrict our discussion to the non-Bayesian approaches taken.

As we use methods designed for convex problems, we start by using a careful initialization of our procedure. This attempts to begin the algorithm close to a global minimum of the objective, so that when we do descend the likelihood, we descend a good one, knowing that local minima exist. This initialization was composed of several steps.

1. Create a proposal 'base' graph from $\hat{y}_{ij} = \sum_k y_{ijk} > 0$
2. Fit a separate single-graph latent space model to each layer
3. Transform the layer estimates to minimize their distance from the base estimate

The third step in this procedure was initially done with a Procrustes transform [?], as is used in Bayesian estimation of Latent Space Models. This transform is used because proposal graphs in a sampling procedure can produce a graph with many different unimportant variations, as the likelihood is invariant to translation, rotation, scaling, and flipping. The Procrustes transform is thus used to remove these transformations, while keeping the variations that are of importance to the likelihood. However this transform minimizes the overall distance between *all* pairs of points in two shapes, and in this problem, we require something slightly different. The goal here is to minimize only distances between positions that correspond to the same node – this is the property that minimizes the ϵ's, and makes the embeddings comparable.

To solve this problem we developed and propose the Anticrustean transformation. A closed form solution, should one exist, has not been found, but a simple implementation combines line searches over the several classes of transforms to which the likelihood is invariant. Together, these steps produce a good initial estimate of all of the models parameters, which are used as starting points in the proximal gradient procedure.

4   Results

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

5   Discussion

6   Conclusion

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

References