Computational Techniques in Latent Variable Network Models

Lily Chou, Ben Draves, Nathan Josephs, Kelly Kung

December 12, 2018

1 Introduction

Game of Thrones is a popular HBO TV series adapted from George R.R. Martin's best-selling book series *A Song of Ice and Fire*. The medieval fantasy epic describes the stories of powerful families - kings and queens, knights and renegades, liars and honest men - playing a deadly game for control of the Seven Kingdoms of Westeros and to sit atop the Iron Throne. Conspiracy and deception, power and exile, blood and tears run through the plot, sewing together characters with various backgrounds including royals and peasants, as well as ice zombies and dragons. As the plot develops with each book release, readers wonder where the storyline leads. Within the Seven Kingdoms, enemies become friends and vice-versa, all the while winter spreads as the battle of ice and far draws nearer. For many characters, it is clear who is good and who is evil, but such moral assignments are skewed by the readers' biases. Here, we propose to put aside our personal feelings and let the data decide.

After discovering a dataset on the exchanges between the characters from the third book, *A Storm of Swords*, we start to wonder if, and what, information we can extract. In particular, how can we make inferences on character being good or bad. To address this, we turn to research of Peter Hoff on latent network models. In Section 2, we show how the relationships between characters from the book naturally arise as a network. Before fitting a model, we explain the latent network model framework in Section 3. We then present two methods for fitting our model in Section 4 and then compare the results of the two methods in Section 5. Finally, in Section 6 we end with a discussion on the implications of our findings, as well as possible future work. We provide our code in the Appendix.

2 Data

Due to its global fame, *Game of Thrones* has been studied in many different contexts, especially in network analysis. Therefore, there are many readily available datasets. In our project, we use the dataset from [**beveridge2016network**],

which contains information about characters' interactions in the third book of the series. In this case, an *interaction* occurs if the characters' names appear within fifteen words of one another. This could mean that the characters interacted with each other, conversed with each other, or were generally mentioned together by another means. There is also a column that contains the number of times each pair interacts with one another. Using this dataset, we constructed a weighted network using the number of interactions as weights. Here, the nodes represent the characters and the edges represent the interactions. We use an adjacency matrix, A, to represent the network, where the $a_{i,j}$ element represents the number of times the characters interacted with each other. Note that this means if $a_{i.j} = 0$, there are no recorded interactions between character i and j based on how an *interaction* is defined. Although the original dataset is intended as a directed network, we treat it as an undirected network in order to simplify our models.

After transforming the dataset, our network G contains $N_V(G) = 107$ nodes and $N_E(G) = 352$ edges which means it is quite sparse since it only contains approximately 6.20% of $\binom{N_V(G)}{2} = 5,671$ possible edges. Figure **include figure** shows the network described. In order to account for the sparsity of our network, we consider a subnetwork which only contains pairs of characters with at least 75 interactions **or more?**. We chose a cutoff of 75 interactions because we want to focus our analysis on only the main characters. Looking at the distribution of the weighted degree, we see that 65.42% of the characters had fewer than 75 interactions. Therefore, it makes sense to use this cutoff to limit our analysis to only the main characters. By doing so, our new network G' contains $N_V(G') = 35$ nodes and $N_E(G') = 140$ edges. Here, we see that the network now contains 23.53% of 595 possible edges, which is a more appropriate level of density for our analysis. Everything that follows is done on this subnetwork G'. Figure **include figure next to other figure** shows this subnetwork G', and indeed, we recognize the main characters remain in our network.

3 Models

3.1 Latent Network Models

Review of Hoff's papers (need's citations):

1. Introduction of network data
2. Community detection in networks
3. Latent variable models
4. Inclusion of observed covariates

3.2 Model Formulation

Following the layout in Section 3.1, we model the presence of an edge given our latent variables as

$$\text{logit } \mathbb{P}(Y_{ij} = 1|Z) = ||Z_i - Z_j|| + \epsilon_{ij}$$

where

$$Z_i \overset{iid}{\sim} \sum_{g=1}^{G} \lambda_g \text{MVN}_d(\mu_g, \sigma_g^2 I_d)$$

Putting priors over μ_g and σ_g^2, as well as introducing the latent variable K representing the group from which Z is drawn, we have the following model formulation:

$$\begin{aligned}
Y_{ij}|Z_i, Z_j &\sim \text{Bern}\Big[\text{logit}^{-1}\big(\|Z_i - Z_j\|\big)\Big] \\
Z_i|K_i = k_i &\overset{iid}{\sim} N(\mu_{k_i}, \sigma_{k_i}^2 I_2) \\
K &\overset{iid}{\sim} \text{Multinoulli}\big(G, \frac{1}{G}\big) \\
\mu_{k_i} &\overset{iid}{\sim} \text{MVN}_2(0, I_2) \\
\sigma_{k_i}^2 &\overset{iid}{\sim} \chi_2^2
\end{aligned}$$

With these, we can write the complete likelihood and log-likelihood, which we will use in both our fitting procedures. We let $\theta = (\mu, \sigma^2, K)$ denote our nuisance parameters. Then, we have

$$\begin{aligned}
\mathcal{L}(Z, \theta; Y) &= \prod_{i<j} \mathbb{P}(Y_{ij}|Z_i, Z_j)\mathbb{P}(Z_i|K_i, \mu_{k_i}, \sigma_{k_i}^2)\mathbb{P}(Z_j|K_j, \mu_{k_j}, \sigma_j^2)\mathbb{P}(K_i)\mathbb{P}(\mu_{k_i})\mathbb{P}(\sigma_{k_i}^2)\mathbb{P}(K_j)\mathbb{P}(\mu_{k_j})\mathbb{P}(\sigma_{k_j}^2) \\
&= \prod_{i<j} \Big(\text{logit}^{-1}\big(\|Z_i - Z_j\|\big)\Big)^{Y_{ij}} \Big(1 - \text{logit}^{-1}\big(\|Z_i - Z_j\|\big)\Big)^{1-Y_{ij}} \\
&\qquad \times \phi(\mu_{k_i}, \sigma_{k_i}^2) \times \frac{1}{G} \times f_{MVN}(0, I_2) \times f_{\chi_2^2} \\
&\qquad \times \phi(\mu_{k_j}, \sigma_{k_j}^2) \times \frac{1}{G} \times f_{MVN}(0, I_2) \times f_{\chi_2^2} \\
&\propto \prod_{i<j} \Big(\text{logit}^{-1}\big(\|Z_i - Z_j\|\big)\Big)^{Y_{ij}} \Big(1 - \text{logit}^{-1}\big(\|Z_i - Z_j\|\big)\Big)^{1-Y_{ij}} \\
&\qquad \times \frac{1}{\sigma_{k_i}} \exp\Big\{ -\frac{1}{2\sigma_{k_i}^2}(Z_i - \mu_{k_i})^2 \Big\} \frac{1}{\sigma_{k_j}} \exp\Big\{ -\frac{1}{2\sigma_{k_j}^2}(Z_j - \mu_{k_j})^2 \Big\} \\
&\qquad \times \exp\Big\{ -\frac{1}{2}\mu_{k_i}^T \mu_{k_i} \Big\} \exp\Big\{ -\frac{1}{2}\mu_{k_j}^T \mu_{k_j} \Big\} \\
&\qquad \times \exp\Big\{ -\frac{\sigma_{k_i}^2}{2} \Big\} \exp\Big\{ -\frac{\sigma_{k_j}^2}{2} \Big\}
\end{aligned}$$

Hence the log-likelihood, up to a constant, is

$$
\begin{aligned}
l(Z, \theta; Y) \stackrel{c}{=} \sum_{i<j} \Big\{ & Y_{ij} \|Z_i - Z_j\| - \log\big(1 + \exp\big(\|Z_i - Z_j\|\big)\big) \Big\} \\
& - \log \sigma_{k_i} - \frac{1}{2\sigma_{k_i}^2}(Z_i - \mu_{k_i})^2 - \log \sigma_{k_j} - \frac{1}{2\sigma_{k_j}^2}(Z_j - \mu_{k_j})^2 \\
& - \frac{1}{2}\Big(\mu_{k_i}^T \mu_{k_i} - \mu_{k_j}^T \mu_{k_j} - \sigma_{k_i}^2 - \sigma_{k_j}^2\Big)
\end{aligned}
$$

4 Computational Methods

4.1 EM

One method for finding the latent variables Z_i for each node is the Expectation-Maximization (EM) algorithm. Unfortunately, the log-likelihood given in Section 3.2 cannot easily be handled with EM. In particular, finding $\mathbb{E}\Big[l(Z, \theta; Y)\Big]_{Z_i, Z_j | Y, \theta}$ is difficult. One solution is to sample and use Monte Carlo estimation in the E-step, then proceed with the M-step. This is the so-called Monte Carlo EM (MCEM) method.

Instead, we simplify our model to make EM more analytically tractable. In Section 4.2, we fit the full model using MCMC. Here, we fit the simplified model using latent distances $d_{ij} \equiv \|Z_i - Z_j\|$

$$
\begin{aligned}
Y_{ij} | Z_i, Z_j &\sim \text{Bern}\Big[\text{logit}^{-1}\big(d_{ij}\big)\Big] \\
d_{ij} &\sim \text{Beta}(\alpha, \beta)
\end{aligned}
$$

Then

$$
\begin{aligned}
\mathcal{L}(d_{ij}, \alpha, \beta; Y) &= \prod_{i<j} \Big(\text{logit}^{-1}\big(d_{ij}\big)\Big)^{Y_{ij}} \Big(1 - \text{logit}^{-1}\big(d_{ij}\big)\Big)^{1 - Y_{ij}} \\
&\quad \times \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} d_{ij}^{\alpha - 1} (1 - d_{ij})^{\beta - 1} \\
l(d_{ij}, \alpha, \beta; Y) &= \sum_{i<j} Y_{ij} \log\Big(\frac{d_{ij}}{1 - d_{ij}}\Big) + (1 - d_{ij}) \\
&\quad + \log\Gamma(\alpha + \beta) - \log\Gamma(\alpha) - \log\Gamma(\beta) \\
&\quad + (\alpha - 1)\log d_{ij} + (\beta - 1)\log(1 - d_{ij})
\end{aligned}
$$

4.1.1 E-Step

do E-step

Algorithm 1: EM for simplified latent network model

```
1 LNM EM (G);
  Input  : Graph G
  Output: Latent Distances d_ij
2 E;
3 M;
```

4.1.2 M-Step

do M-step

4.2 MCMC

Algorithm 2: Gibbs sampler for latent network model

```
1 LNM MCMC (G, params, ns);
  Input  : Graph G
           params
           Number of samples ns
  Output: Posterior p(WHAT)
2 Initialize what;
3 for t = 2, ..., ns do
4 |  sampler
5 end
```

5 Results

1. Analysis
2. Results

6 Conclusion

book 3 is nice because its midpoint of releases so characters have developed, but what if we perform same analysis for the other books and map a character's goodness across each book