Mixed Membership Stochastic Blockmodels (MMSBs) for Network Tomography

Xinyu Kong (xkong8)
Ruoqian Liu (rliu30)
Xuan Wang (xwang174)

May 8, 2017

1 Introduction

Networks are a fundamental form of representations of complex systems in biology, social science, and other fields. As problems arising in these fields, analyzing population of entities such as molecules or individuals interconnected by a set of relationships. Sufficient information can be obtained by studying this kind of networks, such as how individuals organize themselves into groups or how the patterns of social interactions.

In our project, we study statistical inference problem of interpreting dynamic behavior of temporally evolving networks. Our research question and model are based on publication *A state-space mixed membership blockmodel for dynamic network tomography* [1] by Eric P. Xing, Wenjie Fu and Le Song. In real-worlds complex systems, the attributes and relationships of vertices in a network can fluctuate, evolve, emerge and terminate stochastically. The term network tomography is defined as roles or functions undertaken by the nodal entities that determine the edge probability. Our goal is to develop a statistical model and algorithm with which such information can be inferred from networks via posterior inference.

In a social network such as company employee network, network tomography can capture the latent social roles of individuals. Inferring such roles based on the social interactions among individuals is fundamental for modeling behaviors and interpreting social structures. By modeling network tomography, analyzer can discover changing roles among actors in a network.

1.1 Background

There has been a variety of successes in network analysis and there has been progress in statistical modeling of social networks, traditionally focusing on descriptive models such as the exponential random graph models, and recently moving toward latent space models. We will discuss this in details in next section. A major limitation of previous methods for network modeling and inference is that they assume each actor, such as social individuals in a network, undertakes a single and invariant role when interacting with other actors. However,

in realistic social scenarios, actors can play multiple roles and the specific role being played depends on whom the actor is interacting with.

1.2 Model and Study Goal

In this paper, we study a Bayesian approach for network inference that will capture an actors role in dynamic networks. The proposed method is built on a modified version of the mixed membership stochastic blockmodel (MMSB) [2], which enables network links to be realized by role-specific local connection mechanisms. Each vertex is involved in multiple roles, and the proposed model supports analyzing patterns of interactions between actors via inferring an embedding of a network. MMSB model can be extended to dynamic MMSB (dMMSB) model if one wants to capture the dynamics of role evolution of actors, and this model allows one to infer the trajectory of the roles of each actor based on the posterior distribution of role-vector.

Given network data, the MMSB will be learned based on the maximum likelihood principle using EM algorithm [3]. We will illustrate this model and validate the inference algorithms by synthetic networks. We generated three sets of synthetic networks, each of which has 100 individuals and 3 roles. We use 3 different sets of role-vector priors and role-compatibility matrices to mimic different real-life situations. Estimated errors will be shown to evaluate models and algorithms.

In following sections, we will briefly introduce some related work and present the implementation steps of MMSB model in details. In experiments section, we will present case studies on simulated synthetic networks and validation of the model.

2 Related work

The traditional model based statistical analysis of network data are focusing on descriptive models such as the exponential random graph models [4], and moving toward more generative types of models. The mixed membership stochastic blockmodel proposed in Airoldi [2] integrates ideas from these models, but went further by allowing each node to belong to multiple groups. The MMSB model developed ealier has been applied for role identification in Sampsons 18-monk social network and functional prediction in a protein-protein interaction network (PPI) [2]. For each monk, it yields a multi-class social-identity prediction which captures the fact that his interaction with different other monks may be under different contexts. For each protein, it yields a multi-class functional prediction which captures the fact that its interaction with different proteins may be under different functional contexts.

The model intended to use state-space model for inferring underlying functional changes in network entities. This scheme has been adopted in recent works on extracting evolving topical themes in text documents [5] or author embeddings [6].

3 Algorithm and Model Description

Note: This section is based on E. P. Xing, W. Fu and L. Song's paper.[1]

The purpose of the model is to solve roles of each vertex and the compatibility of the roles based on an observed network. Consider a network $G = \{V, E\}$ in which V represent the vertex set of N vertices and $E = \{e_{i,j}\}_{i,j=1}^{N}$ denotes the set of links between N vertices.

Suppose every vertex v_i in the network can undertake K different roles based on a distribution $P(\cdot)$ and the weights of involved roles are represented by a normalized vector $\vec{\pi}_i$ of K dimensions. These roles of the vertices are also called memberships of a latent class. We assume that each vertex can undertake different roles, forming mixed-memberships, and the links between vertices are according to a compatibility function related to the roles undertaken by the vertex-pair. The compatibility coefficients between all possible roles are defined in a $K \times K$ role-compatibility matrix $B = \{\beta_{k,l}\}_{k,l=1}^{N}$.

3.1 Sample modeling

3.1.1 Mixed membership stochastic blockmodel (MMSB)

Network links can be generated by a role-specific local interaction mechanism, in which link between each pair of vertices i and j is dependent on the roles undertaken by i and j. Specifically, suppose vertices i and j undertake roles k and l respectively, then the probability of having a link between them follows a unique probability distribution $P(\cdot|\beta_{k,l})$.

Given the role weights vector $\vec{\pi}_i$ and the role-compatibility matrix B, a basic MMSB can be generated by the following scheme:

1. For each vertix i, draw the mixed-membership vector: $\vec{\pi}_i \sim P(\cdot|\theta)$
2. For each possible interacting vertex j of vertex i, draw the link indicator $e_{i,j} \in \{0, 1\}$ as follows:
   - Draw latent roles of i when it is to interact with j as $\vec{z}_{i\to j} \sim Multinomial(\cdot|\vec{\pi}_i, 1)$, and the latent roles of j when it is approached by i as $\vec{z}_{j\leftarrow i} \sim Multinomial(\cdot|\vec{\pi}_j, 1)$. Here $\vec{z}_{i\to j}$ and $\vec{z}_{j\leftarrow i}$ are unit indictor vectors of K dimensions with only one element, for example the k^{th} element, is 1 and the rest are 0, meaning that it undertakes the k^{th} role .
   - and draw $e_{i,j}|(z_{i\to j,k} = 1, z_{j\leftarrow i,l} = 1) \sim Bernoulli(\cdot|\beta_{k,l})$. It means that the probability of forming a link between i and j when they undertake roles k and l respectively follows Bernoulli distribution given the compatibility coefficient between roles k and l.

MMSB generated in this scheme indicates that the nature and strength of the interaction is controlled by the role-pair and their compatibility. For all the other vertices with the same role, a vertex i is likely to have similar relationship with them.

3.1.2 Logistic-normal MMSB (LNMMSB)

To capture nontrivial correlations among the weights, we employ a logistic-normal distribution over a simplex. It is referred to as a logistic-normal MMSB (LNMMSB).

In the LNMMSB model, the mixed-membership vector $\vec{\pi}_i$ is transformed into the logistic form that:

$$\vec{\pi}_{i,k} = exp\left\{\gamma_{i,k} - C(\vec{\gamma}_i)\right\} \tag{1}$$

where the K dimension vector for each vertex $\vec{\gamma}_i$ is drawn from:

$$\vec{\gamma}_i \sim Normal(\vec{\mu}, \Sigma) \tag{2}$$

and $C(\vec{\gamma}_i)$ is a normalization constant that:

$$C(\vec{\gamma}_i) = log(\sum_{k=1}^{K} \exp\{\gamma_{i,k}\}) \tag{3}$$

and Σ is considered as a known parameter as a $K \times K$ matrix. Due to normalizability constrain of the multinomial parameters, $\vec{\pi}_i$ only has $K-1$ degree of freedom. $\vec{\pi}$ and $\vec{\gamma}$ are both $N \times K$ matrix to represent possible roles (column) for every vertex (row).

3.2 Variational approximation

For static LNMMSB, the goal is to estimate the role-vectors $\vec{\pi}_i$ given parameters $\vec{\mu}$, Σ, B and the observed variables E. It also equals to compute estimates of $\vec{\gamma}_\cdot$ and the role indicators $\vec{z}_{\cdot\to\cdot}$ and $\vec{z}_{\cdot\leftarrow\cdot}$.

As the marginal posterior is

$$\begin{aligned} &p(\vec{\gamma}_\cdot, \vec{z}_{\cdot\to\cdot}, \vec{z}_{\cdot\leftarrow\cdot} | \vec{\mu}, \Sigma, B, E) \\ &\quad \propto \prod_i p(\vec{\gamma}_i | \vec{\mu}, \Sigma) \prod_{i,j} p(\vec{z}_{\cdot\to\cdot}, \vec{z}_{\cdot\leftarrow\cdot} | \vec{\gamma}_i, \vec{\gamma}_j) p(e_{i,j} | \vec{z}_{\cdot\leftarrow\cdot} | \vec{\gamma}_i, \vec{\gamma}_j, B). \end{aligned} \tag{4}$$

It can be approximated with a simpler marginals $q(\cdot) = q_\gamma(\cdot) q_z(\cdot)$, and these marginals approximation are isomorphic to the true conditional distribution given their expected Markov Blanket that:

$$q_\gamma(\vec{\gamma}_i) = p(\vec{\gamma}_i | \vec{\mu}, \Sigma, \left\langle \vec{z}_{i\to\cdot} \right\rangle_{q_z}, \left\langle \vec{z}_{i\leftarrow\cdot} \right\rangle_{q_z}) \tag{5}$$

$$q_z(\vec{z}_{i\to j}, \vec{z}_{j\leftarrow i}) = p(\vec{z}_{i\to j}, \vec{z}_{j\leftarrow i} | e_{i,j}, B, \left\langle \vec{\gamma}_i \right\rangle_{q_\gamma}, \left\langle \vec{\gamma}_j \right\rangle_{q_\gamma}) \tag{6}$$

The optimal marginal distribution will be updated untill the change is neglectable. The update formula is derived as following:

$$q_\gamma(\vec{\gamma}) \sim N(\tilde{\gamma}_i, \tilde{\Sigma}_i) \tag{7}$$

where

$$\tilde{\gamma} = \vec{\mu} + \tilde{\Sigma}(\left\langle \vec{m}_i \right\rangle_{q_z} - (2N-2)\ \vec{g} + (2N-2) H \hat{\gamma}_i - (2N-2) H\ \vec{\mu}\ . \tag{8}$$

$$\tilde{\Sigma} = (\Sigma^{-1} + (2N-2)H)^{-1} \tag{9}$$

and $\vec{g}$ and H are the first and second derivative of $C(\vec{\gamma}) = log(\sum_{k=1}^{K} exp(\gamma_{i,k}))$ in Taylor expansion. $\vec{m}_i$ is a vector that represents the roles of vertex i such that $m_{ik} = \sum_{j \neq i}(z_{i \to j,k} + z_{i \leftarrow j,k})$. This step is to update role possibility vectors in order to sample network models.

While $q_z(\vec{z}_{i \to j}, \vec{z}_{j \leftarrow i})$ is updated as

$$q_z(\vec{z}_{i \to j}, \vec{z}_{j \leftarrow i}) \sim Multinomial(\vec{\delta}_{ij}) \tag{10}$$

where $\vec{\delta}$ is a $N \times N$ matrix, that

$$\delta_{ij(u,v)} = \frac{1}{c} \exp\Big(\langle \gamma_{i,u} \rangle_{q_\gamma} + \langle \gamma_{j,v} \rangle_{q_\gamma}\Big) \beta_{u,v}^{e_{ij}} (1 - \beta_{u,v})^{1 - e_{ij}} \tag{11}$$

and C is the normalization function. $\exp\Big(\langle \gamma_{i,u} \rangle_{q_\gamma} + \langle \gamma_{j,v} \rangle_{q_\gamma}\Big)$ is proportional to $\pi_{i,u}\pi_{j,v}$ which is the possibility that vertices i and j adopt roles u and v. Therefore, $\delta_{ij(u,v)}$ is the possibility of vertices i and j with roles u and v forming the observed relationship. This step is to sample a model given the parameters as described in section 2.1.

The update of $q_\gamma(\vec{\gamma})$ and $q_z(\vec{z}_{i \to j}, \vec{z}_{j \leftarrow i})$ will continue until convergence when the change of log-likelihood is less than 10^-6 in absolute value.

3.3 Parameter estimation

A basic EM-style strategy to estimate the other parameters is: first fix the current parameters of $\vec{\mu}$ and Σ, and compute the role distribution variables $\vec{\gamma}$, B and $\vec{z}$; then re-estimate the prior parameters by maximizing the log-likelihood given the updated variables. Repeating these two-step procedure until convergence will finally get the estimation of the role distribution variables. In addition, to get result with the best likelihood, the procedure can be repeated with different initialization points, and the model with the best log-likelihood are chosen as the solution.

More specifically, to estimate parameters $\vec{\mu}$, Σ, B and role vectors $\vec{\gamma}$ and $\vec{z}_{i \to j}$, $\vec{z}_{j \leftarrow i}$, we first initialize $\vec{\mu}$ and Σ, and use a EM-style procedure to estimate $\vec{\gamma}$ and $\vec{z}_{i \to j}$, $\vec{z}_{j \leftarrow i}$ by updating the distribution with equation (7) and (10) as described in section 2.2 until convergence. Then $\vec{\mu}$ and Σ are updated as:

$$\hat{\mu} = \frac{1}{N} \sum_i \gamma_i \tag{12}$$

$$\hat{\Sigma} = \frac{1}{N} \sum_i \Sigma_i + Cov(\tilde{\gamma}_{1:N}) \tag{13}$$

and to make the role-compatibility matrix closely coupled with role vectors, B matrix is updated together with the role vectors with equation:

$$\hat{\beta_{k,l}} = \frac{\sum_{i,j} e_{ij} \delta_{ij(k,l)}}{\sum_{i,j} \delta_{ij(k,l)}} \tag{14}$$

where data $E = \{e_{ij}\}$ is known.

Therefore, the overall algorithm is summarized as below:

1. initialize $B \sim U[0,1]$, $\vec{\mu}$, $\Sigma = 10I$.
2. while not converge (Outer Loop):
   (a) Initialize $q(\vec{\gamma}_i)$.
   (b) While not converged and iteration less than threshold (Inner Loop):
      i. update $q(\vec{z}_{i\to j}, \vec{z}_{j\leftarrow i})$ based on equation (10) given $\vec{\gamma}$, B and observed network E.
      ii. update $q(\vec{\gamma})$ based on equation (7) given $\vec{z}_{i\to j}$, $\vec{z}_{j\leftarrow i}$, $\vec{\mu}$ and Σ.
      iii. update B based on equation (14) given updated $\vec{\gamma}$, $\vec{\mu}$ and Σ.
   (c) update $\vec{\mu}$ based on equation (12) and $\vec{\Sigma}$ based on equation (13) given updated $\vec{\gamma}$ and observed network interaction matrix E.

4 Experiments

4.1 Synthetic network

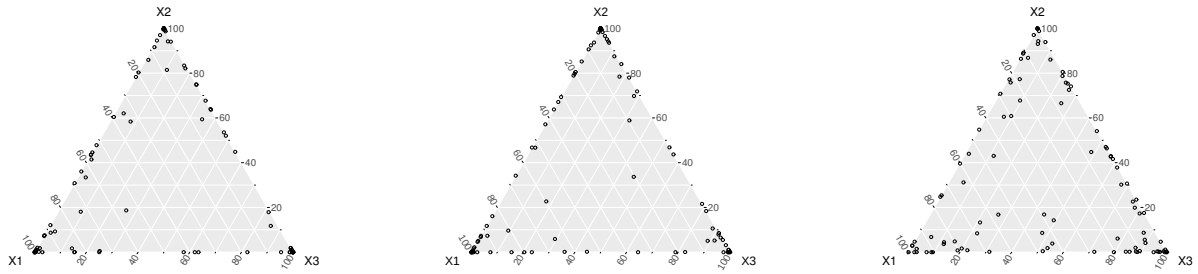
We used three different synthetic networks to test the algorithm according to the paper. There are some real life datasets that can also be used to test the algorithm, e.g. use a gene-gene interaction network to infer the role of each gene. However, due to the data accessibility and time limit, we didn't try the algorithm on the real life datasets. We believe these simple synthetic networks can work well in illustrating the inference accuracy and usefulness of algorithm.

The three synthetic networks each contains 100 vertices and 3 roles for each vertex. The mixed membership vectors of each vertex in the three synthetic networks are shown in Figure 1. The role-compatibility matrix of all the roles in the three synthetic networks are shown in Table 1. The characteristics of each synthetic network are as follow:

1. For network 1, most vertices have only a single role and the role-compatibility matrix is diagonal. It means that each vertex interacts most with the vertices that have the same kind of role with it.

2. For network 2, most vertices still have only a single role, but the role-compatibility matrix contains off-diagonal elements. It means that each vertex have a higher possibility to interact with the vertices that have a different role than that in network 1.

3. For network 3, each vertex can have obvious mixed roles and the within role interactions are much weaker than that in network 1 and 2. This is the most difficult case among the three.



(a) Network 1 (b) Network 2 (c) Network 3

Figure 1: Mixed membership vector of the synthetic networks.

	Role1	Role2	Role3
Role1	1	0	0
Role2	0	1	0
Role3	0	0	1

(a) Network 1

	Role1	Role2	Role3
Role1	1	0.3	0
Role2	0.3	1	0
Role3	0	0	1

(b) Network 2

	Role1	Role2	Role3
Role1	0.45	0	0.05
Role2	0	0.5	0
Role3	0	0	0.4

(c) Network 3

Table 1: Role-compatibility matrix of the synthetic networks.

4.2 Inference accuracy

We run the algorithm on the three synthetic networks and the results are shown as below. The mixed membership vectors of each vertex in the three simulated networks are shown in Figure 2. The role-compatibility matrix of all the roles in the three simulated networks are shown in Table 2.

From the results we could see that the mixed membership vectors are close to what we observed from the synthetic networks. For network 1 and 2, most of the vertices have a single role, which is represented by a dot at each of the three corners. For network 3, there are more vertices that have mixed roles, which is represented by a dot close to the center of the ternary plot.

However, the role-compatibility matrices are not simulated very well. Especially for network 1 and 2, although they have relatively large diagonal values, they still contain some obvious off-diagonal values which indicates an interaction between different roles. We will take a closer look at the distances between the synthetic and simulated networks and the final log likelihoods to get the goodness of fit of the LNMMSB model.

	Role1	Role2	Role3
Role1	0.49	0.29	0.75
Role2	0.46	0.12	0.07
Role3	0.31	0.30	0.20

(a) Network 1

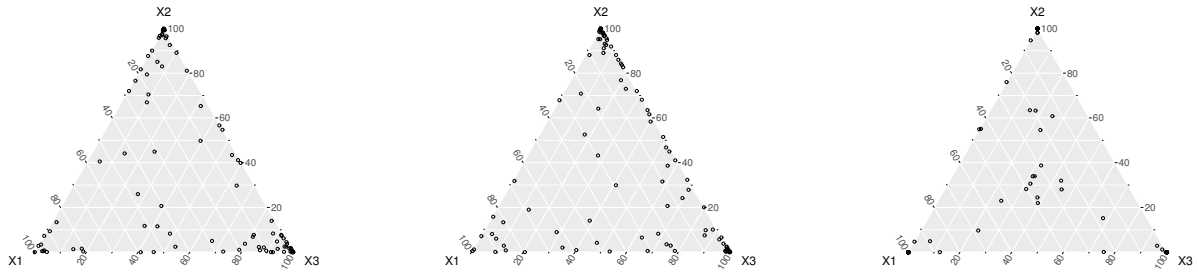
	Role1	Role2	Role3
Role1	0.64	0.29	0.01
Role2	0.19	0.53	0.40
Role3	0.18	0.85	0.66

(b) Network 2

	Role1	Role2	Role3
Role1	0.67	0.01	0.03
Role2	0.16	0.46	0.32
Role3	0.10	0.28	0.06

(c) Network 3

Table 2: Role-compatibility matrix of the simulated networks.



(a) Network 1 (b) Network 2 (c) Network 3

Figure 2: Mixed membership vector of the simulated networks.

4.3 Goodness of fit of LNMMSB

The L1 and L2 distances between the three pairs of synthetic and simulated networks are shown in Figure 3. From the boxplots, we could see that the average L1 distances are around 0.5 and the average L2 distances are around 0.3. From Table 3, we could see a summary of the distances and final log-likelihoods of the simulations of the three networks. The log-likelihoods of network 1 and 2 are around -7000 and -9000, and the log-likelihood of network 3 is around -6000. The result of network 3 simulation is better than the results of network 1 and 2.

From the paper, the reported average L2 distance of network 2 is 0.1 and the loglikelihood is -5691.7. From the comparison we could see that our result is not as good as what is reported in the paper. This could be due to a simplified implementation of our algorithm. However, our model still showed that the it could be used to infer the hidden mixed memberships of each vertex in a network by only observing all the edges of the network.

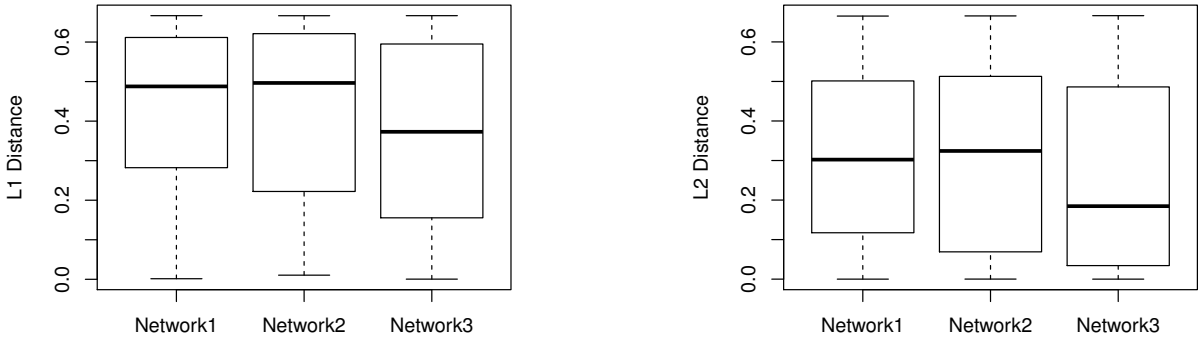


Figure 3: L1 & L2 distances between the synthetic and simulated networks.

	Avg. L1 distance	Avg. L2 distance	Log-likelihood
Network 1	0.42	0.31	-7278.72
Network 2	0.42	0.30	-9017.41
Network 3	0.37	0.25	-6016.38

Table 3: Average L1 & L2 distances and log-likelihood of the simulated networks.

5 Conclusion

According to the results of the simulations on the three synthetic networks, we conclude that our model can infer the hiddern mixed membership vectors of each vertex in a network by only observing all the edges in the network. This model can be very useful in real life cases, e.g., inferring the hiddern roles of each gene in a gene-gene interaction network.

Moreover, this model can be further extended to model dynamic networks. For example, the gene-gene interaction network can evolve over time during different life stages of the fruit-fly. In this model, the logistic normal prior distribution can be adjusted to have time-evolving parameters, thus simulating a time-evolving network. There are also many other things that can be taken into consideration when analyzing a real life network, e.g. higher order structures (motifs) in the network and heterogeneity of a complex network.

6 Reference

References

[1] Eric P. Xing, Wenjie Fu and Le Song. (2010). The Annals of Applied Statistics. *A state-space mixed membership blockmodel for dynamic network tomography.* Vol. 4, No. 2, 535-566.

[2] Airoldi, E. M., Blei, D. M., Fienberg, S.E. and Xing, E.P. (2008). *Mixed membership stochastic blockmodel.* J. Mach. Learn. Res. 9 1981-2014.

[3] Ghahramani, Z. and Beal, M. J. (2001). Propagation algorithms for variational Bayesian learning. *In advances in Neural information Processing Systems 13.* MIT Press, Boston, MA.

[4] Frank, O. and Strauss, D. (1986). *Markov graphs.* J. Amer. Statist. Assoc. 81 832-842. MR0860518

[5] Blei, D. and Lafferty, J. (2006). Correlated topic models. *In advances in Neural Information Processing Systems 18.* MIT Press, Boston, MA.

[6] Sarkar, P. and Moore, A. W. (2005). *Dynamic social network analysis using latent space models.* SIGKDD Explor. Newsl. 7 31-40.

7 Code for modeling

```
# STAT 525 Final Project
# Xinyu Kong (xkong8) Ruoqian Liu (rliu30) Xuan Wang (xwang174)
# State-Space Mixed Membership Blockmodel for Dynamic Network Tomography

######################################################################
library(MASS)
library(numDeriv)
library(gtools)
library(Matrix)

# Logistic Normal Mixed Membership Stochastic Blockmodel (LNMMSB)
LNMMSB <- function(N_rol, N_node, E){
  # Create the function of logistic normalization
  C <- function(g) {
    max(g)+log(sum(exp(g-max(g))))
  }

  # Initialize result
  result <- list()
  result[[1]] <- matrix(0, N_node, N_rol) # P
  result[[2]] <- matrix(0, N_rol, N_rol) # B
  result[[3]] <- -10000 # ll1
  result[[4]] <- matrix(0, N_node, N_rol) # G
  result[[5]] <- matrix(0, N_rol, N_rol) # S1
  result[[6]] <- -10000 # ll0

  # Initialize B, Mu and Sigma
  X <- mvrnorm(N_rol, mu = rep(0, N_rol), Sigma = diag(N_rol))
  B <- pnorm(X)
  M <- mvrnorm(1, mu = rep(0, N_rol), Sigma = diag(N_rol))
  S <- 10*diag(N_rol)

  # Initialize convergence and iteration
  # Here we set the convergence condition as the change of log-likelyhood <=
     10^(-6) based on the paper.
  ll0 <- -10000 # This is set based on the paper results section
  delta <- 1
  iteration <- 0
  counter <- 0
  while(delta >= 10^(-6) & (counter < 10)){
    # Initialize Gamma and calculate Pi distribution
    G <- mvrnorm(N_node, mu = M, Sigma = S)
    # while(delta >= 10^(-6)){
    while((delta >= 10^(-6)) & (iteration < 10)){
      # iteration number is not mentioned in the paper
```

```
# Update Z
P <- matrix(0, N_node, N_rol)
Z <- vector()
for (i in 1:N_node){
  P[i, ] <- exp(G[i, ] - C(G[i, ]))
  Z <- rbind(Z, t(rmultinom(1, 1, P[i, ])))
}
# Calculate the log-likelyhood given the input network E
# Calculate the log-likelyhood by importance sampling?
D <- matrix(0, N_node, N_node)
B1 <- matrix(0, N_rol, N_rol)
B2 <- matrix(0, N_rol, N_rol)
m <- matrix(0, N_node, N_rol)
Delta <- matrix(0, N_node, N_node)
for (i in 1:N_node) {
  for (j in 1:N_node){
    x <- which(Z[i,]==max(Z[i,]))
    y <- which(Z[j,]==max(Z[j,]))
    p <- B[x, y]
    D[i, j] <- dbinom(E[i, j], 1, p)
    # Calculate delta[i,j]
    Delta[i, j] <- (P[i, x]*P[j, y])*(p^E[i, j])*((1-p)^(1-E[i, j]))
    B1[x, y] <- B1[x, y] + E[i,j]*Delta[i,j]
    B2[x, y] <- B2[x, y] + Delta[i, j]
    if (j != i){
      m[i, x] <- m[i, x] + Delta[i, j]
      m[i, y] <- m[i, y] + Delta[i, j]
    }
  }
}
# Log-likelihood calculation?
ll1 <- sum(log(D))
print(ll1)
if (ll1 > result[[3]]) {
  result[[1]] <- P
  result[[2]] <- B
  result[[3]] <- ll1
  result[[4]] <- G
  result[[5]] <- S1
  result[[6]] <- ll0
}
# Update delta
delta <- abs(ll1 - ll0)
ll0 <- ll1
# Update Gamma and B
G1 <- vector()
S1 <- matrix(0, N_rol, N_rol)
```

```
  for (i in 1:N_node) {
    g <- G[i, ]
    Deriv1 <- grad(C, g)
    Deriv2 <- hessian(C, g)
    s <- solve(solve(S) + (2*N_node-2)*Deriv2)
    s <- matrix(nearPD(s)$mat, N_rol, N_rol)
    g1 <- t(M+s%*%(m[i,]+(2*N_node-2)*(-Deriv1+Deriv2%*%g-Deriv2%*%M)))
    G1 <- rbind(G1, mvrnorm(1, mu = g1, Sigma = s))
    S1 <- S1 + s
  }
  G <- G1
  B <- B1/B2
  # Update iteration
  iteration <- iteration + 1
  print(iteration)
}
# Update Mu and Sigma
G <- result[[4]]
S1 <- result[[5]]
B <- result[[2]]
ll0 <- result[[6]]

M <- colMeans(G)
S <- S1/N_node + cov(G)
counter <- counter + 1
iteration <- 0
print(counter)
# Calculate log-likelyhood again!
G <- mvrnorm(N_node, mu = M, Sigma = S)
P <- matrix(0, N_node, N_rol)
Z <- vector()
for (i in 1:N_node){
  P[i, ] <- exp(G[i, ] - C(G[i, ]))
  Z <- rbind(Z, t(rmultinom(1, 1, P[i, ])))
}
# Calculate the log-likelyhood given the input network E
# Calculate the log-likelyhood by importance sampling?
D <- matrix(0, N_node, N_node)
for (i in 1:N_node) {
  for (j in 1:N_node){
    x <- which(Z[i,]==max(Z[i,]))
    y <- which(Z[j,]==max(Z[j,]))
    p <- B[x, y]
    D[i, j] <- dbinom(E[i, j], 1, p)
  }
}
# Log-likelihood calculation?
```

```
    ll1 <- sum(log(D))
    print(ll1)
    # Update delta
    delta <- abs(ll1 - ll0)
  }
  return(result)
}


######################################################################

# Logistic Normal Mix Membership Stochastic Blockmodel (LNMMSB) practice

######################################################################

# Synthetic Network Construction
library(ggtern)

N_node <- 100
N_rol <- 3

# Network 1
B_N1 <- cbind(c(1, 0, 0), c(0, 1, 0), c(0, 0, 1))

# M_N1 <- c(0, 0, 0)
# S_N1 <- 10*diag(N_rol)
# G_N1 <- mvrnorm(N_node, mu = M_N1, Sigma = S_N1)
# P_N1 <- exp(G_N1-log(rowSums(exp(G_N1))))
# ternplot_N1 <- ggtern(data=data.frame(P_N1),
#                       aes(x = X1, y = X2, z = X3))
# ternplot_N1 + geom_point(size=1)

P_N1 <- rbind(rdirichlet(33, c(0.8, 0.1, 0.1)),
              rdirichlet(33, c(0.1, 0.8, 0.1)),
              rdirichlet(34, c(0.1, 0.1, 0.8)))
P_N1 <- P_N1[sample(nrow(P_N1)),]
P_N1 <- P_N1/rowSums(P_N1)
ternplot_N1 <- ggtern(data=data.frame(P_N1),
                      aes(x = X1, y = X2, z = X3))
ternplot_N1 + geom_mask() + geom_point(size=1, pch=1)

Z_N1 <- vector()
for (i in 1:N_node){
  Z_N1 <- rbind(Z_N1, t(rmultinom(1, 1, P_N1[i, ])))
}
E_N1 <- matrix(0, N_node, N_node)
for (i in 1:N_node) {
```

```
  for (j in 1:N_node){
    x <- which(Z_N1[i,]==max(Z_N1[i,]))
    y <- which(Z_N1[j,]==max(Z_N1[j,]))
    p <- B_N1[x, y]
    E_N1[i, j] <- rbinom(1, 1, p)
  }
}

# Network 2
B_N2 <- cbind(c(1, 0.3, 0), c(0.3, 1, 0), c(0, 0, 1))

P_N2 <- rbind(rdirichlet(33, c(0.8, 0.1, 0.1)),
              rdirichlet(33, c(0.1, 0.8, 0.1)),
              rdirichlet(34, c(0.1, 0.1, 0.8)))
P_N2 <- P_N2[sample(nrow(P_N2)),]
P_N2 <- P_N2/rowSums(P_N2)
ternplot_N2 <- ggtern(data=data.frame(P_N2),
                      aes(x = X1, y = X2, z = X3))
ternplot_N2 + geom_mask() + geom_point(size=1, pch=1)

Z_N2 <- vector()
for (i in 1:N_node){
  Z_N2 <- rbind(Z_N2, t(rmultinom(1, 1, P_N2[i, ])))
}
E_N2 <- matrix(0, N_node, N_node)
for (i in 1:N_node) {
  for (j in 1:N_node){
    x <- which(Z_N2[i,]==max(Z_N2[i,]))
    y <- which(Z_N2[j,]==max(Z_N2[j,]))
    p <- B_N2[x, y]
    E_N2[i, j] <- rbinom(1, 1, p)
  }
}

# Network 3
B_N3 <- cbind(c(0.45, 0, 0), c(0, 0.5, 0), c(0.05, 0, 0.4))

P_N3 <- rbind(rdirichlet(33, c(0.6, 0.2, 0.2)),
              rdirichlet(33, c(0.2, 0.6, 0.2)),
              rdirichlet(34, c(0.2, 0.2, 0.6)))
P_N3 <- P_N3[sample(nrow(P_N3)),]
P_N3 <- P_N3/rowSums(P_N3)
ternplot_N3 <- ggtern(data=data.frame(P_N3),
                      aes(x = X1, y = X2, z = X3))
ternplot_N3 + geom_mask() + geom_point(size=1, pch=1)

Z_N3 <- vector()
```

```
for (i in 1:N_node){
  Z_N3 <- rbind(Z_N3, t(rmultinom(1, 1, P_N3[i, ])))
}
E_N3 <- matrix(0, N_node, N_node)
for (i in 1:N_node) {
  for (j in 1:N_node){
    x <- which(Z_N3[i,]==max(Z_N3[i,]))
    y <- which(Z_N3[j,]==max(Z_N3[j,]))
    p <- B_N3[x, y]
    E_N3[i, j] <- rbinom(1, 1, p)
  }
}

######################################################################

# Inference Accuracy (Figure 2)
LNMMSB_N1 <- LNMMSB(N_rol, N_node, E_N1)
LNMMSB_N2 <- LNMMSB(N_rol, N_node, E_N2)
LNMMSB_N3 <- LNMMSB(N_rol, N_node, E_N3)

LNMMSB_ternplot_N1 <- ggtern(data=data.frame(LNMMSB_N1[[1]]),
                     aes(x = X1, y = X2, z = X3))
LNMMSB_ternplot_N1 + geom_mask() + geom_point(size=1, pch=1)
LNMMSB_ternplot_N2 <- ggtern(data=data.frame(LNMMSB_N2[[1]]),
                   aes(x = X1, y = X2, z = X3))
LNMMSB_ternplot_N2 + geom_mask() + geom_point(size=1, pch=1)
LNMMSB_ternplot_N3 <- ggtern(data=data.frame(LNMMSB_N3[[1]]),
                   aes(x = X1, y = X2, z = X3))
LNMMSB_ternplot_N3 + geom_mask() + geom_point(size=1, pch=1)

# B matrix of the three simulated networks
print(LNMMSB_N1[[2]])
print(LNMMSB_N2[[2]])
print(LNMMSB_N3[[2]])

######################################################################

# Goodness of fit of LNMMSB and MMSB (Figure 3 and Table 1)
# Figure 3
# L1 distance calculation
LNMMSB_L1_N1 <- vector()
for (i in 1:N_node){
  LNMMSB_L1_N1 <- c(LNMMSB_L1_N1, mean(abs(P_N1[i, ]-LNMMSB_N1[[1]][i, ])))
}
LNMMSB_L1_N2 <- vector()
for (i in 1:N_node){
  LNMMSB_L1_N2 <- c(LNMMSB_L1_N2, mean(abs(P_N2[i, ]-LNMMSB_N2[[1]][i, ])))
```

```
}
LNMMSB_L1_N3 <- vector()
for (i in 1:N_node){
  LNMMSB_L1_N3 <- c(LNMMSB_L1_N3, mean(abs(P_N3[i, ]-LNMMSB_N3[[1]][i, ])))
}

# L2 distance calculation
LNMMSB_L2_N1 <- vector()
for (i in 1:N_node){
  LNMMSB_L2_N1 <- c(LNMMSB_L2_N1, mean((P_N1[i, ]-LNMMSB_N1[[1]][i, ])^2))
}
LNMMSB_L2_N2 <- vector()
for (i in 1:N_node){
  LNMMSB_L2_N2 <- c(LNMMSB_L2_N2, mean((P_N2[i, ]-LNMMSB_N2[[1]][i, ])^2))
}
LNMMSB_L2_N3 <- vector()
for (i in 1:N_node){
  LNMMSB_L2_N3 <- c(LNMMSB_L2_N3, mean((P_N3[i, ]-LNMMSB_N3[[1]][i, ])^2))
}

# Boxplots
boxplot(cbind(LNMMSB_L1_N1, LNMMSB_L1_N2, LNMMSB_L1_N3),
        names = c('Network1', 'Network2', 'Network3'),
        ylab = 'L1 Distance')
boxplot(cbind(LNMMSB_L2_N1, LNMMSB_L2_N2, LNMMSB_L2_N3),
        names = c('Network1', 'Network2', 'Network3'),
        ylab = 'L2 Distance')


# Table 1
mean(LNMMSB_L1_N1)
mean(LNMMSB_L1_N2)
mean(LNMMSB_L1_N3)
mean(LNMMSB_L2_N1)
mean(LNMMSB_L2_N2)
mean(LNMMSB_L2_N3)
LNMMSB_N1[[3]]
LNMMSB_N2[[3]]
LNMMSB_N3[[3]]

######################################################################

# Goodness of fit of LNMMSB

######################################################################

# Drawing pairwise plot of the true and simulated networks
```

```
pairtern <- function(P1, P2){
  N1 <- data.frame(P1)
  N2 <- data.frame(P2)
  N1$group <- 'True'
  N2$group <- 'Simulated'
  df <- rbind(N1, N2)
  pair <- cbind(setNames(N1[, 1:3], c('X', 'Y', 'Z')),
                setNames(N2[, 1:3], c('Xend', 'Yend', 'Zend')))
  plot <- ggtern(df, aes(x = X1, y = X2, z = X3))
  plot +
    geom_mask() +
    geom_point(size=1, aes(shape=group)) +
    scale_shape_manual(values=c(4, 1)) +
    geom_segment(aes(x = X, y = Y, z = Z,
                     xend = Xend, yend = Yend, zend = Zend),
                 size=0.1,
                 data = pair)
}

pairtern(P_N1, LNMMSB_N1[[1]])
pairtern(P_N2, LNMMSB_N2[[1]])
pairtern(P_N3, LNMMSB_N3[[1]])
```