ST 790 Assignment 4

David Elsheimer and Jimmy Hickey

11/12/2020

Instruction

This assignment consists of 3 problems. Choose 2 out of the 3 problems. The assignment is due on **Friday, November 12** at 11:59pm EDT. Please submit your assignment electronically through the **Moodle** webpage. The assignment can be done as a group with at most 3 members per group (please include the name of the group members on the front page of the assignment).

Problem 1

Skim the following article. In particular take a look at Algorithm 1 in the referenced paper. Next download the **DBLP4** dataset from here. Now try to analyze this dataset using the Algorithm 1 (SVM-Cone) form the referenced paper (see also section 4.1.2 of the referenced paper). For the evaluation metric, use rank correlation (see section 4.1.2. of the referenced paper).

Note This problem might be a tad tricky. It is perfectly fine if your results are not as good as that presented in the paper. As long as you are not getting a performance that is worse than chance, it should be fine.

```
A_edge <- as.matrix(read.delim("DBLP4_adjacency.txt", header=FALSE)) #edgelist
A_graph <- graph.data.frame(A_edge, directed=FALSE) #graph object
A<- as_adjacency_matrix(A_graph) #Adjacency matrix
#A_f <- norm(as.matrix(A), type="F") #frobenius norm

theta<- as.matrix(read.delim("DBLP4_community.txt", sep= " ", header=FALSE))

colnames(theta) <- c("row", "community", "val")

decomp<- spectrum(A_graph, which=list(pos="LM", howmany=3)) #500 works!
eigval <- decomp$values
eigvec <- decomp$vectors

#sqrt(sum(eigval^2))/A_f #Needs to be >.8

Zhat <- eigvec

###Need to perform row normalization
rownorm <- function(i){
  (Zhat[i,]-mean(Zhat[i,]))/sd(Zhat[i,])
  }

Yhat <- t(sapply(1:nrow(Zhat), rownorm))
```

```r
model <- svm(Yhat, type='one-classification', kernel="linear")

w <- t(model$coefs) %*% model$SV
b <- -model$rho

Yhat_w <- Yhat %*% t(w)

delta <- 1
filtered <- Yhat_w[(Yhat %*% t(w) < b)==TRUE]

which((Yhat %*% t(w) < b)==TRUE)[c(3,232,925)]
```

```
## [1]    7  332 1960
```

```r
##clustering

k3 <- kmeans(filtered, centers = 3, nstart = 25)

###Need a point from each cluster
#k3$cluster
#lets just choose 1, 925, 232

###Feeding that back into Yhat to get M...

Yhat_c <- Yhat[c(3,232,925),]
Mhat <- Zhat %*% t(Yhat_c)%*%solve(Yhat_c%*%t(Yhat_c)+0.00000001)

#theorem 3.2
#need proper indices again
which((Yhat %*% t(w) < b)==TRUE)[c(3,232,925)]
```

```
## [1]    7  332 1960
```

```r
Vhat_c <- Zhat[c(7, 332, 1960),]
Nhat_c <- t(sapply(1:nrow(Vhat_c), rownorm))

D <-sqrt(diag(Nhat_c%*%Vhat_c%*%diag(eigval)%*% t(Vhat_c)%*%t(Nhat_c)))
Dhat <- diag(D)

Fhat <-  Mhat%*%Dhat%*%as.matrix(rep(1,3))


thetahat <- solve(Diagonal(length(Fhat),Fhat))%*%Mhat%*%Dhat

theta_empty <- Matrix(0,nrow(thetahat),3)


thetaval_j <- function(j){
  #theta_empty[which(theta[,2]==j),j] <-
  temp <- theta[which(theta[,2]==j),c(1,3)]
  theta_empty[temp[,1],j] <- temp[,2]
  theta_empty
}

theta_empty <- thetaval_j(1)
theta_empty <- thetaval_j(2)
theta_empty <- thetaval_j(3)

theta_proper <- theta_empty

avgrankcorr <- function(sigma){
  summand <- cor(thetahat[,1], theta_proper[,sigma[1]], method="spearman")
  summand <- c(summand,cor(thetahat[,2], theta_proper[,sigma[2]], method="spearman"))
  summand <- c(summand,cor(thetahat[,3], theta_proper[,sigma[3]], method="spearman"))
  mean(summand)
}


sigmas <- list(c(1,2,3),
               c(2,3,1),
               c(3,1,2),
               c(1,3,2),
               c(2,1,3),
               c(3,2,1))
sums <- sapply(sigmas,avgrankcorr)
```

Using the code above $\hat{\Theta}$ was generated using algorithm 1 and additional definitions as laid out in the paper. Based on how rank correlation is defined in the paper, the average rank correlation between $\hat{\Theta}, \Theta$ here is 0.0464011, which corresponds to the permuation c(3, 2, 1).

Problem 2

Skim the following article (maybe only the first 16 pages, unless you really have time and care about the theory). Now take a look at section 4.1 on model selection for stochastic blockmodels. Next try to reproduce Table 1 (or a part of Table 1) but only for the ECV algorithm (Algorithm 3) with L_2 loss; you don't need to consider L_2 loss with stability.

```r
library(randnet)
```

```
## Warning: package 'randnet' was built under R version 4.0.3

## Loading required package: entropy

## Warning: package 'entropy' was built under R version 4.0.3

## Loading required package: AUC

## Warning: package 'AUC' was built under R version 4.0.3

## AUC 0.3.0

## Type AUCNews() to see the change log and ?AUC to get an overview.
```

```r
library(irlba)
```

```
## Warning: package 'irlba' was built under R version 4.0.3
```

```r
# step 1 of alg 3
# this will be looped for m = 1, ... , N
step1 = function(A, m, n, p, Kmax){

  ####
  # step a
  ###

  # split into train and test with probability p
  # We only want to choose from the upper (or lower) triangular
  # vertices, since we will be adding the lower (or upper) triangular part
  # i.e. if node pair (i,j) is selected, then (j,i) will be added after

  # list of pairs in the upper triangular portion of the training set
  # we are looking at n*(n-1)/2 vertices because we are not considering
  # the diagonal

  train_pairs = rbinom(n*(n-1)/2, 1, p)



  A_train = matrix(0, nrow = n, ncol = n)
  A_test = matrix(0, nrow = n, ncol = n)

  A_train[upper.tri(A_train, diag=FALSE)] =
    A[upper.tri(A,diag=FALSE)]*train_pairs

  # add lower triangular parts
  A_train = A_train +  t(A_train)

  # sanity check
  # isSymmetric(A_train)

  # A_test is all indices not in A_train
  A_test[upper.tri(A_test, diag=FALSE)] =
    A[upper.tri(A,diag=FALSE)]*(1 - train_pairs)
  A_test = A_test + t(A_test)

  # sanity check
  # isSymmetric(A_test)


  ###
  # step b
  ###

  L = matrix(0, ncol = 2, nrow = Kmax)

  for(k in 1:Kmax){
    L[k,] = step1b(k, n, p, A_train, A_test, train_pairs)
  }

  return(as.vector(L))

}



# step 1b of alg 3
# Thank you TK for helping us with this step!
# We needed A LOT of help.
step1b = function(k, n, p, A_train, A_test, train_pairs){
  ###
  # step 1b i
  ###

  # rank k constrained matrix completion
  # estimate k values with rank constraint k
  irlb_out = irlba(A_train/p, nu = k, nv = k)
  Ahat_k = irlb_out$u %*% diag(irlb_out$d, nrow = k) %*% t(irlb_out$v)


  ###
  # step 1b ii
  ###

  # spectral clustering on Ahat_k

  Xhat_k = irlba(Ahat_k, nu=k, nv=k)$u

  # if k = 1, then we have an SBM
  # if k > 1, then we have a DCSBM
  # as the algorithm says, we need to perform
  # spectral and spherical clustering on them respectively

  if(k == 1){
    # spectral clustering
    X_norm = Xhat_k / outer(sqrt(rowSums(Xhat_k^2)),c(1))
  } else{
    # speherical clustering
    X_norm = Xhat_k / outer( sqrt(rowSums(Xhat_k^2)),rep(1:k))
  }

  # kmeans freaks out if you have NaNs in the data
  X_norm[is.nan(X_norm)]=rep(0,k)


  # now do K-means
  clust1 = kmeans(Xhat_k, centers = k , iter.max = 100)
  clust2 = kmeans(X_norm, centers = k, iter.max = 100)

  chat1 = clust1$cluster
```

```r
chat2 = clust2$cluster

###
# step 1 b iii
###

# estimate the model probability matrices
# follows from the paper, section 3.2

# we will need a matrix of the clusters
clust_mat = diag(k)

# need as.matrix for the case of k = 1
clust1_mat = as.matrix(clust_mat[chat1,], ncol = k)
clust2_mat = as.matrix(clust_mat[chat2,], ncol = k)

# equation 6
# build nhat and Bhat
# also build Ohat since we are iterating over the same indices
nhat = matrix(0, ncol = k, nrow = k)
Ohat = matrix(0, ncol = k, nrow = k)
Bhat = matrix(0, ncol = k, nrow = k)

for(i in 1:k){
  for(j in 1:k){

    if(i == j){
      # length of indicators
      nhat[i,j] = (length(chat1[chat1 == i]) -1) * length(chat1[chat1==j])
    } else{
      nhat[i,j] = length(chat1[chat1 == i]) * length(chat1[chat1==j])
    }

    Bhat[i,j] = sum(A_train * (clust1_mat[,i] %*% t(clust1_mat[,j])))/nhat[i,j]
    Ohat[i,j] = sum(A_train * (clust1_mat[,i] %*% t(clust1_mat[,j])))
  }
}

# calculate theta_hat from eqn at top of page 10

if(k == 1){
  # Ohat will be a vector, so don't need to sum over it
  # have to make the O scalar a vector or R yells at me
  theta2_hat = rowSums(A_train) / as.vector(Ohat)
}  else{
  theta2_hat =rowSums(A_train) / rowSums(Ohat[chat2,])
}

# calculate P from eqn at top of page 10
Phat_1 = clust1_mat %*% Bhat %*% t(clust1_mat)
Phat_2 = (theta2_hat %*% t(theta2_hat)) * Ohat[chat2, chat2] / p
```

```r
  # now we can calculate loss!
  # we need to find the estimates of the Phat matrices
  # that correspond the to the test set
  # this follows the same process as creating the original train/test
  P1_test = matrix(0, nrow = n, ncol = n)
  P2_test = matrix(0, nrow = n, ncol = n)

  P1_test[upper.tri(P1_test, diag=FALSE)] =
    Phat_1[upper.tri(Phat_1,diag=FALSE)]*(1 - train_pairs)
  P1_test = P1_test + t(P1_test)

  P2_test[upper.tri(P2_test, diag=FALSE)] =
    Phat_2[upper.tri(Phat_2,diag=FALSE)]*(1 - train_pairs)
  P2_test = P2_test + t(P2_test)

  loss1 = sum((A_test - P1_test)^2)
  loss2 = sum((A_test - P2_test)^2)


  return(c(loss1, loss2))
}


# alg 3 from Li, Levina, Zhu
# in order to make table 1, it should return
# K, n, lambda, beta, L2 looss
alg3 = function(K, n, lambda, beta, Kmax=5, p=0.90, N=200){

  # need to generate graph each time with new parameters
  graph = BlockModel.Gen(lambda=lambda,
                         n=n,
                         K=K,
                         beta=beta,
                         rho=p )


  # input adjacency matrix
  A = graph$A

  loss_mat = matrix(0, nrow = N, ncol = 2 * Kmax)

  ###
  # step 1
  ###
  for(m in 1:N){
    loss_mat[m,] = step1(A, m, n, p, Kmax)
  }


  ###
  # step 2
  ###
  # qhat this gives the optimal model
  # 1 is SBM and 2 is DCSBM
  qhat = rep(0, N)

  # khat gives the optimal number of communities
  khat = rep(0, N)

  for(m in 1:N){
    # find the index with the lowest and divide by Kmax
    # this is because we have all of the SBM result at the front of the row
    # and all of the DCSBM results at the end of the row
    qhat[m] = ceiling(which.min(loss_mat[m,]) / Kmax)

    #  mod by Kmax to find which community number it is under
    # since there are two columns representing each community number
    # one column for SBM and one for DCSBM
    # add one to Kmax so that a community number of KMax doesn't mod to 0
    khat[m] = which.min(loss_mat[m,])%%(Kmax+1)

  }


  # loss is number of times DCSBM is chosen AND max communities are chosen
  # divided by N
  loss = sum( (qhat == 2) & (khat == K))/N

  return(list(K = K,
              n = n,
              lambda = lambda,
              beta = beta,
              loss = loss))
}





results = data.frame(matrix(vector(), 0, 5,
                dimnames=list(c(), c("K", "n", "lambda", "beta", "L2 loss"))),
                stringsAsFactors=F)


# results = rbind(results, alg3(3, 600, 15, 0.2, Kmax=6, p=0.90, N=200))
# results = rbind(results, alg3(3, 600, 20, 0.2, Kmax=6, p=0.90, N=200))
# results = rbind(results, alg3(3, 600, 30, 0.2, Kmax=6, p=0.90, N=200))
# results = rbind(results, alg3(3, 600, 40, 0.2, Kmax=6, p=0.90, N=200))
#
# results = rbind(results, alg3(5, 600, 15, 0.2, Kmax=6, p=0.90, N=200))
# results = rbind(results, alg3(5, 600, 20, 0.2, Kmax=6, p=0.90, N=200))
# results = rbind(results, alg3(5, 600, 30, 0.2, Kmax=6, p=0.90, N=200))
# results = rbind(results, alg3(5, 600, 40, 0.2, Kmax=6, p=0.90, N=200))
#
```

```r
# results = rbind(results, alg3(5, 1200, 15, 0.2, Kmax=6, p=0.90, N=200))
# results = rbind(results, alg3(5, 1200, 20, 0.2, Kmax=6, p=0.90, N=200))
# results = rbind(results, alg3(5, 1200, 30, 0.2, Kmax=6, p=0.90, N=200))
# results = rbind(results, alg3(5, 1200, 40, 0.2, Kmax=6, p=0.90, N=200))
#
#
#
# results = rbind(results, alg3(3, 600, 40, 0.1, Kmax=5, p=0.90, N=200))
# results = rbind(results, alg3(3, 600, 40, 0.2, Kmax=5, p=0.90, N=200))
# results = rbind(results, alg3(3, 600, 40, 0.5, Kmax=5, p=0.90, N=200))
#
# results = rbind(results, alg3(5, 600, 40, 0.1, Kmax=5, p=0.90, N=200))
# results = rbind(results, alg3(5, 600, 40, 0.2, Kmax=5, p=0.90, N=200))
# results = rbind(results, alg3(5, 600, 40, 0.5, Kmax=5, p=0.90, N=200))
#
# results = rbind(results, alg3(5, 1200, 40, 0.1, Kmax=5, p=0.90, N=200))
# results = rbind(results, alg3(5, 1200, 40, 0.2, Kmax=5, p=0.90, N=200))
# last = rbind(results, alg3(5, 1200, 40, 0.5, Kmax=5, p=0.90, N=200))


load("q2_fullresults.Rdata")



knitr::kable(full_results)
```

	K	n	lambda	beta	loss
1	3	600	15	0.2	0.125
2	3	600	15	0.2	0.145
3	3	600	20	0.2	0.105
4	3	600	30	0.2	0.125
5	3	600	40	0.2	0.135
6	5	600	15	0.2	0.325
7	5	600	20	0.2	0.385
8	5	600	30	0.2	0.270
9	5	600	40	0.2	0.115
10	5	1200	15	0.2	0.090
11	5	1200	15	0.2	0.100
12	5	1200	20	0.2	0.440
13	5	1200	30	0.2	0.630
14	5	1200	40	0.2	0.540
21	3	600	40	0.1	0.035
31	3	600	40	0.2	0.155
42	3	600	40	0.5	0.040
41	5	600	40	0.1	0.030
51	5	600	40	0.1	0.040
61	5	600	40	0.2	0.100
71	5	600	40	0.5	0.005
81	5	1200	40	0.1	0.180
91	5	1200	40	0.2	0.245
101	5	1200	40	0.5	0.090
22	3	600	40	0.1	0.035
32	3	600	40	0.2	0.155

	K	n	lambda	beta	loss
43	3	600	40	0.5	0.040
411	5	600	40	0.1	0.030
52	5	600	40	0.1	0.040
62	5	600	40	0.2	0.100
72	5	600	40	0.5	0.005
82	5	1200	40	0.1	0.180
92	5	1200	40	0.2	0.245
102	5	1200	40	0.5	0.090

Problem 3

Let $\mathbf{A}_1$ and $\mathbf{A}_2$ be two-blocks stochastic blockmodel graphs, with block probability matrices

$$\mathbf{B}_1 = \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}, \quad \mathbf{B}_2 = \begin{bmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{bmatrix}$$

Suppose for simplicity that the first $n/2$ vertices of both $\mathbf{A}_1$ and $\mathbf{A}_2$ are assigned to block 1 and that the last $n/2$ vertices of both $\mathbf{A}_1$ and $\mathbf{A} - 2$ are assigned to block 2. Assume that $\mathbf{A}_1(i,j)$ and $\mathbf{A}_2(k,\ell)$ are independent of one another if $\{i,j\} \neq \{k,\ell\}$. Finally, $\mathbf{A}_1(i,j)$ and $\mathbf{A}_2(i,j)$ are correlated with correlation $\rho \in [-1,1]$. We assume that ρ is the same for all $\{i,j\}$ pairs.

- Given $\mathbf{A}_1$ and $\mathbf{A}_2$, formulate a test statistic for testing $\mathbb{H}_0 : \rho = 0$ against the alternative hypothesis that $\mathbb{H}_1 : \rho \neq 0$. In other words, test the hypothesis that $\mathbf{A}_1$ is independent of $\mathbf{A}_2$.
- Do you think your test procedure is valid and consistent as $n \to \infty$? (assuming that the parameters of $\mathbf{B}_1$ and $\mathbf{B}_2$ and ρ are kept constant).
- How would you adapt this procedure when the block assignments are unknown, or, in addition, if the graphs involved are degree corrected SBMs instead of the vanilla SBMs ?