R codes for Mixture of networks based on penalized composite likelihood

Jangsun Baek, Dept. of Statistics, Chonnam National University, South Korea
2022. 10. 08.

IMPORTANT input data preparation for the first version of R code

Categorical response of the jth categorical variable, x_j, of an observation in data set must be an integer in 1 to r_j, j = 1, ..., p. That is, categorical responses of each variable should be transformed to sequential integers starting from 1 to r_j. For example, if the response of X_1, X_2, X_3 belongs to one of two, three, and two categories, respectively, then x_1 is either 1 or 2, x_2 is either 1, 2, or 3, and x_3 is either 1 or 2. Therefore, the input data of (x_1, x_2, x_3) would be (1, 1, 1), (1, 2, 1), (2, 3, 1), (2, 3, 2), but neither (1, 4, 6) nor (3, 1, 3). Future versions of R code will accept any type of reponses of categorical variables.

##

1. Packages needed for Mixture of networks

```
> install.packages("poLCA")  # LCA for obtaining initial parameters
> library("poLCA")
> install.packages("tmvnsim")  # for package "qgraph"
> install.packages("psych")  # for package "qgraph"
> install.packages("qgraph")  # plotting cluster network graphs
> library(qgraph)
```

2. R codes

Load and source the following functions.

2.1 "mnpcl": clustering program of mixture of networks

```
#
# mnpcl(X, g, n_level, lambda, initmax, itmax, tol_logL, init_para = NULL)
```

#
input:
X: categorical data matrix (nxp)
g: number of components (clusters)
n_level: vector of the number of response levels for variables: (r_1, r_2, ..., r_p)
lambda: tuning parameter of penalty (must be less than max(obs_freq)), where obs_freq[max(n_level),max(n_level),p,p] is the observed frequency in contingency tables
initmax: number of initializations
itmax: maximum number of EM iterations
tol_logL: tolerance of convergence
#
#
output(list):
$pivec: vector of pi_i
$logL: loglikelihood
$cluster: vector of clustered labels in (1,2,..., g)
$theta: bivariate joint prob. penalized mle
$theta_tilde: bivariate joint prob. mle
$thetacond_tilde: conditional prob. mle

2.2 "est_mnpcl": parameter estimation using EM algorithm
2.3 "mstep_mnpcl": M-step
2.4 "tau_mnpcl": posterior prob. estimation
2.5 "logL_mnpcl": penalized composite log-likelihood
2.6 "logLpred_mnpcl": predicted composite log-likelihood
2.7 "sim_data_generation": synthetic data generation
2.8 "next_data_generation": for "sim_data_generation"
2.9 "error.rate": clustering error calculation

3. Zoo dataset, true class labels, and number of response levels

>load("stuff.RData") # Zoo data and other R objects used in the following examples
3.1 **xnew_zoo**: Zoo data
3.2 **zoo_level**: the vector of the number of response levels
3.3 **zoonew_label**: true class labels of xnew_zoo

4. Examples

4.1 Synthetic data

4.1.1 Generation of synthetic data

```
# theta_cond: conditional probability of X_2 given X_1, X_3 given X_2, and X_4 given X_3 for two groups
> theta_marg1<-matrix(c(.01,.99,.99,.01),c(2,2)) # marginal probability of X_1 for two groups
> sim_level <- c(2,3,3,2)
> xx<-sim_data_generation(100, 4, 2, sim_level, theta_marg1, theta_cond)
> xx_label<-c(rep(1,100),rep(2,100)) # true group labels
> xx1<-xx[,,1]
> xx2<-xx[,,2]
> xx<-rbind(xx1,xx2)
> xx<-data.frame(xx)
> colnames(xx)<-c("x1","x2","x3","x4")
> perm_sim<-sample(1:200,replace=FALSE)

# synthetic data matrix and true labels
> x_sim<-xx[perm_sim,]  # 200 four-dimensional synthetic data (200x4)
> sim_label<-xx_label[perm_sim] # true group label vector
```

4.1.2 Clustering the synthetic data into two groups with lambda = 0.6

```
> result_sim<-mnpcl(X=x_sim, g=2, n_level=sim_level, lambda=0.6, initmax=10, itmax=500, tol_logL=1.e-3, init_para = NULL)
initialization =  1
initialization =  2
initialization =  3
initialization =  4
initialization =  5
initialization =  6
initialization =  7
initialization =  8
initialization =  9
initialization =  10
n_g = 114 # number of observations allocated into cluster 1
n_g = 86 # number of observations allocated into cluster 2
```

```
> ACC <- 1-error.rate(result_sim$cluster, sim_label) # ACC of MN-PCL
[1] 0.92 # The ACC may be different because the synthetic data is generated randomly.
```

4.2 Clustering Zoo data into seven groups with lambda = 0.2

```
> result<-mnpcl(xnew_zoo, 7, zoo_level, 0.2, 10, 500, 1.e-3, init_para = NULL)
initialization =  1
initialization =  2
initialization =  3
initialization =  4
initialization =  5
initialization =  6
initialization =  7
initialization =  8
initialization =  9
initialization =  10
n_g = 14
n_g = 4
n_g = 10
n_g = 8
n_g = 37
n_g = 7
n_g = 21
> ACC <- 1-error.rate(result$cluster, zoonew_label)
[1] 0.8910891
> str(result)
List of 7
 $ pivec           : num [1:7] 0.1295 0.0447 0.0963 0.0875 0.3656 ...
 $ theta           : num [1:6, 1:6, 1:16, 1:16, 1:7] NA NA NA NA NA NA NA NA ...
 $ logL            : num -16344
 $ cluster         : int [1:101] 1 1 1 3 7 5 4 7 5 5 ...
 $ theta_tilde     : num [1:6, 1:6, 1:16, 1:16, 1:7] 1 NA NA NA NA NA NA 0 NA  ...
 $ thetacond_tilde: num [1:6, 1:6, 1:16, 1:16, 1:7] NA NA NA NA NA NA NAA NA ...
 $ call            : language mnpcl(X = xnew_zoo, g = 7, n_level = zoo_level, lambda = 0.2, initmax = 10, itmax = 500, tol_logL = 0.001, init_para = NULL)
 - attr(*, "class")= chr "mnpcl"
>
```

5. Plotting cluster network graphs (Figure 3 of Zoo data)

5.1 Load and source functions.

```
> source('D:/A_array.R')
> source('D:/make_adj.R')
> source('D:/generate_plot.R')
```

5.2 Select the indices of variables to be plotted in cluster network graph.

```
> node_select<-c(1,2,4,5,6,9,13) # X_1, X_2, X_4, X_5, X_6, X_9, X_13
```

5.3 Set the number of response levels of all variables.

```
> n_level <- c(rep(2,12),6,rep(2,3)) # 2 for X_1, 2 for X_2, ..., 2 for X_12, 6 for X_13, 2 for X_14, ..., 2 for X_16
> n_level
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 6 2 2 2
```

5.4 Converting input data to an adjacency array

```
#
# A_array(X, n_level)
#
# input:
#     X: categorical data matrix (nxp)
#    n_level: vector of the number of response levels for variables: (r_1, r_2, ..., r_p)
# output:
#    A(max(n_level),max(n_level), p, p, n): adjacency array
#

> A <- A_array(xnew_zoo, n_level)
> str(A)
```

5.5 Make adjacency matrices for clusters

```
#
# make_adj(A, cluster_label, n_level, node_select , g)
#
# input:
#    A: adjacency array (output of "A_array")
#    cluster_label:  vector of clustered labels in (1,2,..., g) (output of "mnpcl")
#    n_level: vector of the number of response levels for variables
#    node_select: the indices of variables to be plotted in cluster network graph
```

```
#    g: the number of clusters
#
# output(list):
#    $result_list: adjacency frequency matrices for clusters
#    $margi_val: marginal weights of nodes of variables for each cluster
#    $select_name: indexed node names of the selected variables
#    $groups: colors of nodes in graph

> g <- 7 # number of clusters = 7 for Zoo data
> final_result <- make_adj(A, result$cluster, n_level, node_select, g)
> names(final_result)
[1] "result_list" "margi_val"    "select_name" "groups"

> result_list <- final_result$result_list
> margi_val <- final_result$margi_val
> select_name <- final_result$select_name
> groups <- final_result$groups
```

5.6 Plotting network graphs: “generate_plot” function

```
#
# generate_plot(result_list, groups, save=F, save_path = " ", file_type="pdf")
#
# input:
#    result_list: adjacency frequency matrices for clusters from “make_adj”
#    groups: colors of nodes in graph from “make_adj”
#
# output:
#    network graphs of the clusters
#    If you want to save each plot, then, set the argument of save to T (save=T).

> generate_plot(result_list, groups, save=T, save_path = "D:/", file_type="pdf")
Output stored in D:/graph1.pdf
Output stored in D:/graph2.pdf
Output stored in D:/graph3.pdf
Output stored in D:/graph4.pdf
Output stored in D:/graph5.pdf
Output stored in D:/graph6.pdf
Output stored in D:/graph7.pdf

# The default is not to save the graphs.
> generate_plot(result_list,groups)
```