Manual˙for˙program

Hui Zhou

Division of Biostatistics, School of Public Health, University of Minnesota

email: zhoux292@umn.edu

Nov 23, 2009

R_common_diag_cov.R and *C_common_diag_cov.cpp* tries to fit penalized model based likelihood to the data assuming common diagonal covariance matrix across different clusters.

- Since the core algorithm is carried out through calling the C++ program from R, the first step is to compile the C++ code. Type at the command prompt in Linux or Unix terminal:

  ```
  R CMD SHLIB C_common_diag_cov.cpp
  ```

  which would generate a *C_common_diag_cov.so* file. Now, read the function into R by typing the following in the R prompt:

  ```
  source("R_common_diag_cov.R")
  ```

- Next the function is run using the command:

```
common_diag_cov(Y,Y_tune,n,n_tune,k,k0,TRUE_INDEX,BIC,num_cluster,lambda)
```

Y	Training data of size $n \times k$, with each row represents one observation.
Y_tune	Tuning data of size $n\_tune \times k$
n	Number of observations for training data
n_tune	Number of observations for tuning data
k	Dimension for both training and tuning data
k0	First k0 variables are informative; if this quantity is unknown, then z_1 and z_2 in the output should be ignored
TRUE_INDEX	The index showing the true group membership of each observation; if this quantity is unknown, then RI and aRI in the output should be ignored
BIC	Boolean variable, if TRUE then select number of cluster, penalty parameters based on BIC, otherwise, use tuning data to select them. Default is FALSE
num_cluster	A vector containing number of clusters to be considered. Default is from 1 to 10
lambda	A vector containing penalty coefficients of the mean vector. Default is from 0 to 20, with step size 1
MAX_iter	Maximum iteration allowed the EM algorithm. Default is 100
threshold	Threshold for convergence. Default is 0.0001

where each argument means:

Therefore when knowing which variables are informative, the user should move those to the left most k0 columns of Y and Y_tune.

The function will return several R objects, which can be assigned to a variable. For example, with all default options, to save the results in the variable out, type the command:

```
out = common_diag_cov(Y,Y_tune,n,n_tune,k,k0,TRUE_INDEX,BIC,
num_cluster,lambda,MAX_iter,threshold)
```

To see the results, use the "$" operator (VariableName$ObjectName). *common_diag_cov*() returns the following objects:

- Example:

num_cluster	optimal number of clusters
lambda	optimal penalty parameter on the mean structure
group_member	posterior probability for each observation belonging to each cluster
z_1	number of estimated non-informative variables among truly informative variables
z_2	number of estimated non-informative variables among truly non-informative variables
RI	rand index between the estimated group membership and TRUE_INDEX
aRI	adjusted rand index between the estimated group membership and TRUE_INDEX

In this example, we first generate a dataset consist of two clusters, with the same diagonal covariance structure but with different mean vector in the first 21 variables, while the remaining 279 variables are noninformative, i.e simulated from standard Normal distribution. The details of how to generate the data is shown below. After data generation, call the function and save the results:

```
out = common_diag_cov(Y,Y_tune,n,n_tune,k,k0,TRUE_INDEX,BIC=T,
    num_cluster=seq(1:5),lambda=seq(0:20),MAX_iter=100,threshold=1e-4))
```

Then to see the output:

```
> out$num_cluster
[1] 2
> out$lambda
[1] 11
> out$z_1
[1] 0
> out$z_2
[1] 277
> out$RI
[1] 1
> out$aRI
```

```
[1] 1
> out$group_member
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [75] 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

The entire example programs is as follows:

```
source("R_CS_diag_cov.R")
################# parameters #####################
set.seed(2)
k0<-21     #  number of attributes defining the clustering
k<- 300   #  number of attributes total
n<- 100    #  total number of objects
n0<-20    #  number of objects in the small cluster
n_tune <-n*1    #  total number of tuning objects
n0_tune <-n0*1  # number of tuning objects in the smaller cluster
nsim=1    # number of simulated data
u1=0.0       #the mean of the first cluster
sd1=1.0       #standard dev
sd2=1.0
u2=0.0 # the mean of one cluster other than the other mean cluster
du=2
du_tune=du
################ generate data ##################
TRUE_INDEX <- c(rep(1,(n-n0)),rep(2,n0))
#cluster structure of the dataset is as following:
#                       informative#   noise#
#                            k0         k-k0
```

```
#cluster 1     n-n0  |       Y11         Y12
#cluster 2     n0    |       Y21         Y22
Y11<-matrix(rnorm(k0*(n-n0),u1,sd1),nrow=n-n0,ncol=k0) # simulated
Y12<-matrix(rnorm((k-k0)*(n-n0),u1,sd1),nrow=n-n0,ncol=k-k0) # simulated
Y21<-matrix(rnorm(n0*k0,u2,sd2), nrow=n0,ncol=k0)+du
Y22<-matrix(rnorm( n0*(k-k0),u2,sd2), nrow=n0,ncol=k-k0)
Y<-rbind(cbind(Y11,Y12),cbind(Y21,Y22))
#
Y11<-matrix(rnorm(k0*(n_tune-n0_tune),u1,sd1),nrow=n_tune-n0_tune,ncol=k0)
Y12<-matrix(rnorm((k-k0)*(n_tune-n0_tune),u1,sd1),nrow=n_tune-n0_tune,ncol=k-k0)
Y21<-matrix(rnorm(n0_tune*k0,u2,sd2), nrow=n0_tune,ncol=k0)+du_tune
Y22<-matrix(rnorm(n0_tune*(k-k0),u2,sd2), nrow=n0_tune,ncol=k-k0)
Y_tune<-rbind(cbind(Y11,Y12),cbind(Y21,Y22))
#
mu_Y<-apply(Y,2,mean)
Y<-t(t(Y)-mu_Y)
Y_tune<-t(t(Y_tune)-mu_Y)
sd_Y<-apply(Y,2,sd)
Y<-t(t(Y)/sd_Y)
Y_tune<-t(t(Y_tune)/sd_Y)
#########################################
out <- common_diag_cov(Y,Y_tune,n,n_tune,k,k0,TRUE_INDEX,BIC=T,
num_cluster=seq(1:5),lambda=seq(0:20),MAX_iter=100,threshold=1e-4)
out$num_cluster
out$lambda
out$z_1
out$z_2
```

```
out$RI

out$aRI

out$group_member
```