Simulation LDA

Sung Bongjung

2021 8 26

In this note, we provide the code for simulation of LDA for breast cancer winsconsin diagonstic dataset. Since it takes a long time for the simulation of our propsosed estimator, approximately 6 hours, we shall provide only the data and simulation code. The rule for LDA and scheme is provided in the previous note.

```
#dataset

wdbc=read.csv("wdbc.csv",header=FALSE)
true_status=wdbc[,1]

#1: case, 0: control
true_status[which(true_status=="M")]=1; true_status[which(true_status=="B")]=0
true_status=as.numeric(true_status)
wdbc=as.matrix(wdbc[,-1])
head(wdbc)
```

```
##         V2    V3     V4     V5      V6      V7     V8      V9    V10     V11
## [1,] 17.99 10.38 122.80 1001.0 0.11840 0.27760 0.3001 0.14710 0.2419 0.07871
## [2,] 20.57 17.77 132.90 1326.0 0.08474 0.07864 0.0869 0.07017 0.1812 0.05667
## [3,] 19.69 21.25 130.00 1203.0 0.10960 0.15990 0.1974 0.12790 0.2069 0.05999
## [4,] 11.42 20.38  77.58  386.1 0.14250 0.28390 0.2414 0.10520 0.2597 0.09744
## [5,] 20.29 14.34 135.10 1297.0 0.10030 0.13280 0.1980 0.10430 0.1809 0.05883
## [6,] 12.45 15.70  82.57  477.1 0.12780 0.17000 0.1578 0.08089 0.2087 0.07613
##         V12    V13   V14    V15      V16     V17     V18     V19     V20
## [1,] 1.0950 0.9053 8.589 153.40 0.006399 0.04904 0.05373 0.01587 0.03003
## [2,] 0.5435 0.7339 3.398  74.08 0.005225 0.01308 0.01860 0.01340 0.01389
## [3,] 0.7456 0.7869 4.585  94.03 0.006150 0.04006 0.03832 0.02058 0.02250
## [4,] 0.4956 1.1560 3.445  27.23 0.009110 0.07458 0.05661 0.01867 0.05963
## [5,] 0.7572 0.7813 5.438  94.44 0.011490 0.02461 0.05688 0.01885 0.01756
## [6,] 0.3345 0.8902 2.217  27.19 0.007510 0.03345 0.03672 0.01137 0.02165
##           V21   V22   V23    V24    V25    V26    V27    V28    V29    V30
## [1,] 0.006193 25.38 17.33 184.60 2019.0 0.1622 0.6656 0.7119 0.2654 0.4601
## [2,] 0.003532 24.99 23.41 158.80 1956.0 0.1238 0.1866 0.2416 0.1860 0.2750
## [3,] 0.004571 23.57 25.53 152.50 1709.0 0.1444 0.4245 0.4504 0.2430 0.3613
## [4,] 0.009208 14.91 26.50  98.87  567.7 0.2098 0.8663 0.6869 0.2575 0.6638
## [5,] 0.005115 22.54 16.67 152.20 1575.0 0.1374 0.2050 0.4000 0.1625 0.2364
## [6,] 0.005082 15.47 23.75 103.40  741.6 0.1791 0.5249 0.5355 0.1741 0.3985
##          V31
## [1,] 0.11890
## [2,] 0.08902
## [3,] 0.08758
## [4,] 0.17300
## [5,] 0.07678
## [6,] 0.12440
```

```
#simulation code

lda_cv=function(true_status,samp,partition){

  n=length(true_status); p=ncol(samp)
  error_samp=numeric(length=partition);error_gl=numeric(length=partition)
  error_mpp=numeric(length=partition);error_map=numeric(length=partition)

  step.size=100
  tol=0.002
  P=matrix(1,p,p)
  diag(P)=0
  lam=0.06

  case=which(true_status==1); control=which(true_status==0)

  for(i in 1:partition){

    case_train=sort(sample(case,size=72,replace=FALSE))
    case_test=sort(setdiff(case,case_train))

    control_train=sort(sample(control,size=119,replace=FALSE))
    control_test=sort(setdiff(control,control_train))

    train=sort(c(case_train,control_train))
    test=sort(c(case_test,control_test))

    status_train=true_status[train]
    status_test=true_status[test]

    samp_train=samp[train,]
    samp_test=samp[test,]

    s1_train=samp[case_train,]; s0_train=samp[control_train,]

    #mean of class 1
    mu1=numeric(length=p)
    for(j in 1:72){
      mu1=mu1+as.numeric(s1_train[j,])
    }
    mu1=mu1/72

    #mean of class 0
    mu0=numeric(length=p)
    for(l in 1:119){
      mu0=mu0+as.numeric(s0_train[l,])
    }
    mu0=mu0/119

    #LDA rule for sample covariance matrix
    est_cov_samp=1/191*t(samp_train)%*%(diag(191)-1/191*rep(1,191)%*%t(rep(1,191)))%*%samp_train+0.1^3*
    est_prec_samp=solve(est_cov_samp)
```

```
#LDA rule for graphical lasso
samp_cov=1/191*t(samp_train)%*%(diag(191)-1/191*rep(1,191)%*%t(rep(1,191)))%*%samp_train+0.001*diag
est_cov_gl=spcov(Sigma=samp_cov,S=samp_cov,lambda=lam*P,step.size=step.size,n.inner.steps=200,
             thr.inner=0,tol.outer=tol,trace=1)$Sigma
est_prec_gl=solve(est_cov_gl)

#LDA rule for propsed estimator for both MPP and MAP
#generate 12000 posterior sample after 3000 burn-in.
mcmc_samp=mcmc(191,samp_cov,1,191*0.05,0.1,12000,3000,0.65)
mcmc_mpp=mpp(mcmc_samp)
mcmc_map=mmp(mcmc_samp)

est_cov_mpp=bcd_covariance2(samp_cov,191,1,191*0.05,10000,0.1^3,mcmc_mpp)
est_prec_mpp=solve(est_cov_mpp)

est_cov_map=bcd_covariance2(samp_cov,191,1,191*0.05,10000,0.1^3,mcmc_map)
est_prec_map=solve(est_cov_map)

#proportion of class j among train set.
p1=72/191
p0=119/191

error_temp_samp=0
error_temp_gl=0
error_temp_proposed_mpp=0
error_temp_proposed_map=0

for(ind in 1:378){
  est_samp=0; est_gl=0l est_mpp=0; est_map=0

  ob=samp_test[ind,]
  rule1_samp=as.numeric(t(ob)%*%est_prec_samp%*%mu1-1/2*t(mu1)%*%est_prec_samp%*%mu1+log(p1))
  rule0_samp=as.numeric(t(ob)%*%est_prec_samp%*%mu0-1/2*t(mu0)%*%est_prec_samp%*%mu0+log(p0))

  rule1_gl=as.numeric(t(ob)%*%est_prec_gl%*%mu1-1/2*t(mu1)%*%est_prec_gl%*%mu1+log(p1))
  rule0_gl=as.numeric(t(ob)%*%est_prec_gl%*%mu0-1/2*t(mu0)%*%est_prec_gl%*%mu0+log(p0))

  rule1_mpp=as.numeric(t(ob)%*%est_prec_mpp%*%mu1-1/2*t(mu1)%*%est_prec_mpp%*%mu1+log(p1))
  rule0_mpp=as.numeric(t(ob)%*%est_prec_mpp%*%mu0-1/2*t(mu0)%*%est_prec_mpp%*%mu0+log(p0))

  rule1_map=as.numeric(t(ob)%*%est_prec_map%*%mu1-1/2*t(mu1)%*%est_prec_map%*%mu1+log(p1))
  rule0_map=as.numeric(t(ob)%*%est_prec_map%*%mu0-1/2*t(mu0)%*%est_prec_map%*%mu0+log(p0))

  if(rule1_samp>rule0_samp){
    est_samp=1
  }
  if(rule1_gl>rule0_gl){
    est_gl=1
  }
  if(rule1_mpp>rule0_mpp){
    est_mpp=1
  }
  if(rule1_map>rule0_map){
```

```
        est_map=1
      }

      diff_samp=est_samp-status_test[ind]
      error_temp_samp=error_temp_samp+abs(diff)

      diff_gl=est_gl-status_test[ind]
      error_temp_gl=error_temp_gl+abs(diff)

      diff_mpp=est_mpp-status_test[ind]
      error_mpp_samp=error_temp_mpp+abs(diff)

      diff_map=est_map-status_test[ind]
      error_temp_map=error_temp_map+abs(diff)
    }
    error_samp[i]=error_temp_samp/378
    error_gl[i]=error_temp_gl/378
    error_mpp[i]=error_temp_mpp/378
    error_map[i]=error_temp_map/378
  }
  error=list(error_samp,error_gl,error_mpp,error_map)
  return(error)
}

#error
error=lda_cv(true_status,wdbc,10)

#sample covariance matrix
mean(error[[1]]); sd(error[[1]])
#0.07698413; 0.01933812

#graphical lasso
mean(error[[2]]); sd(error[[2]])
# 0.07222222; 0.01670605

#proposed (MPP)
mean(error[[3]]); sd(error[[3]])
#0.06613757; 0.01635521

#proposed (MAP)
mean(error[[4]]); sd(error[[4]])
#0.06613757; 0.01635521
```