CCLE data analysis 예제

2024-03-25

Markdown에 사용된 R code와 code에 대한 설명은 CodeExample/Example_CCLE.R 파일에서 확인하실 수 있습니다. 이 예제에서는 Section 4의 CCLE data를 고려합니다.

함수 불러오기

```
path <- "C:/Users/USER/Desktop/Transferlearning/Function/"
source(paste0(path,"Functions_naiveapproaches.R")) # load r function
source(paste0(path,"Functions_MSDtrans.R")) # load r function
source(paste0(path,"Functions_FSDtrans.R")) # load r function
source(paste0(path,"Functions_transSCAD.R")) # load r function
```

데이터 불러오기

```
path <- "C:/Users/USER/Desktop/Transferlearning/data"
dataname <- "CCLEdataset.RDATA"
load(paste(path,dataname,sep="/"))

path <- "C:/Users/USER/Desktop/Transferlearning/data"
dataname <- "CCLEdataset_trainingtest.RDATA"
load(paste(path,dataname,sep="/"))
```

6개의 방법 적합

```
lamLassoseq <- c(0.001, 0.01, 0.05, 0.1,0.15,0.2, 0.3,0.35, 0.4, 0.5, 0.6, 0.7,  0.8, 0.9, 1, 1,1, 1.2, 1.5, 1.8,
2, 2.2 ,2.5)
cv.nuclear(Y=CCLEdataset_trainingtest[[1]]$Ytraining,
           X=CCLEdataset_trainingtest[[1]]$Xtraining,
           B=NULL,
           L=NULL,
           eta=1,lamseq = lamLassoseq,
           tol=1e-04,maxiter=100) -> cvLasso

p <- ncol(CCLEdataset_trainingtest[[1]]$Xtraining)
q <- ncol(CCLEdataset_trainingtest[[1]]$Ytrainin)
dLasso <- svd(matrix(cvLasso$B[-1,],p,q))$d # corresponding singular value
predLasso <- cbind(1,CCLEdataset_trainingtest[[1]]$Xtest) %*% cvLasso$B
Lassoerr <- mean( (predLasso-CCLEdataset_trainingtest[[1]]$Ytest)^2 ,na.rm=TRUE) # MSPE

lamseq_w <- lamLassoseq
cv.pooledNR(Y=CCLEdataset_trainingtest[[1]]$Ytraining,
            X=CCLEdataset_trainingtest[[1]]$Xtraining,
            auxYlist = CCLEdataset$auxYlist,
            auxXlist = CCLEdataset$auxXlist,
            B=NULL,
            L=NULL,eta=1,lamseq = lamLassoseq,
            tol=1e-04,maxiter=100) -> PooledNRres

dPooledNR <- svd(matrix(PooledNRres$B[-1,],p,q))$d # corresponding singular value
predPooledNR <- cbind(1,CCLEdataset_trainingtest[[1]]$Xtest) %*% PooledNRres$B
PooledNRerr <- mean( (predPooledNR-CCLEdataset_trainingtest[[1]]$Ytest)^2 ,na.rm=TRUE) # MSPE

lam_delta <- c(lamseq_w,3)
cv.twostep(Y=CCLEdataset_trainingtest[[1]]$Ytraining,
                          X=CCLEdataset_trainingtest[[1]]$Xtraining,
                          B=NULL, L=NULL,eta=1,
                          lamseq_w = lamseq_w,
                          lamseq_delta=lam_delta,
                          tol=1e-04,maxiter=100,
                          auxYlist=CCLEdataset$auxYlist,
                          auxXlist=CCLEdataset$auxXlist) -> TwosteptransRes

dtwostep <- svd(matrix(TwosteptransRes$B[-1,],p,q))$d # corresponding singular value
predTwostep <- cbind(1,CCLEdataset_trainingtest[[1]]$Xtest) %*% TwosteptransRes$B
Twosteperr <- mean( (predTwostep-CCLEdataset_trainingtest[[1]]$Ytest)^2 ,na.rm=TRUE) # MSPE

library(doParallel)
```

```
## 필요한 패키지를 로딩중입니다: foreach
```

```
## 필요한 패키지를 로딩중입니다: iterators
```

```
## 필요한 패키지를 로딩중입니다: parallel
```

```
cl <- makeCluster(3)
registerDoParallel(cl) # for fsd, msd
MSDtrans(Y=CCLEdataset_trainingtest[[1]]$Ytraining,
                     X=CCLEdataset_trainingtest[[1]]$Xtraining,
                     B=NULL, L=NULL,eta=1,
                     lamseq_w = lamseq_w,
                     lamseq_delta=lam_delta,
                     tol=1e-04,maxiter=100,
                     auxYlist=CCLEdataset$auxYlist,
                     auxXlist=CCLEdataset$auxXlist,
                     nfold=5,nfold_choiceforC=3,
                     nfold_selectionstep_pe=3,
         C=c(0.001, 0.01,  0.05,  0.1)) -> MSDRes

dMSD <- svd(matrix(MSDRes$B[-1,],p,q))$d # corresponding singular value
predMSD<- cbind(1,CCLEdataset_trainingtest[[1]]$Xtest) %*% MSDRes$B
MSDerr <- mean( (predTwostep-CCLEdataset_trainingtest[[1]]$Ytest)^2 ,na.rm=TRUE) # MSPE

FSDtrans(Y=CCLEdataset_trainingtest[[1]]$Ytraining,
         X=CCLEdataset_trainingtest[[1]]$Xtraining,
         B=NULL, L=NULL,eta=1,
         lamseq_w = lamseq_w,
         lamseq_delta=lam_delta,
         tol=1e-04,maxiter=100,
         auxYlist=CCLEdataset$auxYlist,
         auxXlist=CCLEdataset$auxXlist,
         nfold=5,nfold_choiceforC=3,
         nfold_selectionstep_pe=3,
         C=c(0.001, 0.01,  0.05,  0.1)) -> FSDRes

dFSD <- svd(matrix(FSDRes$B[-1,],p,q))$d # corresponding singular value
predFSD <- cbind(1,CCLEdataset_trainingtest[[1]]$Xtest) %*% FSDRes$B
FSDerr <- mean( (predFSD-CCLEdataset_trainingtest[[1]]$Ytest)^2 ,na.rm=TRUE) # MSPE
stopCluster(cl)
stopImplicitCluster()

lamseq_w_SCAD <-  c(0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 0.6, 0.8, 0.9, 1, 1.2, 1.5, 2, 2.5,3,3.5)
lamseq_delta_SCAD <- c(lamseq_w, 5, 7, 10, 20, 50, 70, 100, 200, 500)
sourceindex <- FSDRes$detectedsources
auxYlist_SCAD <- auxXlist_SCAD <- list()
for(ss in 1:length(sourceindex)){
  auxYlist_SCAD[[ss]] <- CCLEdataset$auxYlist[[sourceindex[ss]]]
  auxXlist_SCAD[[ss]] <- CCLEdataset$auxXlist[[sourceindex[ss]]]
}
FSDSCADres <- BIC.TransSCAD(Y=CCLEdataset_trainingtest[[1]]$Ytraining,
                            X=CCLEdataset_trainingtest[[1]]$Xtraining,
                           auxYlist=auxYlist_SCAD,
                           auxXlist=auxXlist_SCAD,
                           lamseq_w=lamseq_w_SCAD,
                           lamseq_delta=lamseq_delta_SCAD,
                           eta=1, a=3.7,
                           B=NULL,L=NULL,Delta=NULL,H=NULL,Pi=NULL,
                           maxiter_inital=300, maxiter_biascorrection=300,
                           tol_inital=1e-04, tol_biascorrection=1e-04,
                           standardize=T)

dFSDSCAD <- svd(matrix(FSDSCADres$B[-1,],p,q))$d # corresponding singular value
predFSDSCAD <- cbind(1,CCLEdataset_trainingtest[[1]]$Xtest) %*% FSDSCADres$B
FSDSCADerr <- mean( (predFSDSCAD-CCLEdataset_trainingtest[[1]]$Ytest)^2 ,na.rm=TRUE) # MSPE
```

예측오차 비교

```
round(c(Lassoerr, PooledNRerr, Twosteperr, MSDerr, FSDerr, FSDSCADerr),3)
```

```
## [1] 0.496 0.568 0.511 0.511 0.441 0.467
```

랭크 비교

```
computerank <- function(x,tol=1e-02){
  sum(x>1e-03)
}
c(computerank(dLasso), computerank(dPooledNR),
  computerank(dtwostep), computerank(dMSD), computerank(dFSD),
  computerank(dFSDSCAD))
```

```
## [1] 2 4 5 5 3 2
```

모형 적합 3번 반복

```
nrep <- 3
errorlist <- list() # to save error
ranklist <- list() # to save rank
Blist <- list() # to save estimates
lamLassoseq <- c(0.001, 0.01, 0.05, 0.1,0.15,0.2, 0.3,0.35, 0.4, 0.5, 0.6, 0.7,  0.8, 0.9, 1, 1,1, 1.2, 1.5, 1.8,
2, 2.2 ,2.5)
lamseq_w <- lamLassoseq
lamseq_delta <- c(lamLassoseq,3)
p <- ncol(CCLEdataset_trainingtest[[1]]$Xtraining)
q <- ncol(CCLEdataset_trainingtest[[1]]$Ytrainin)
cl <- makeCluster(3)
registerDoParallel(cl) # for fsd, msd

lamseq_w_SCAD <-  c(0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 0.6, 0.8, 0.9, 1, 1.2, 1.5, 2, 2.5,3,3.5)
lamseq_delta_SCAD <- c(lamseq_w, 5, 7, 10, 20, 50, 70, 100, 200, 500)
for(i in 1:nrep){
  cv.nuclear(Y=CCLEdataset_trainingtest[[i]]$Ytraining,
             X=CCLEdataset_trainingtest[[i]]$Xtraining,
             B=NULL,
             L=NULL,
             eta=1,lamseq = lamLassoseq,
             tol=1e-04,maxiter=100) -> cvLasso

  cv.pooledNR(Y=CCLEdataset_trainingtest[[i]]$Ytraining,
              X=CCLEdataset_trainingtest[[i]]$Xtraining,
              auxYlist = CCLEdataset$auxYlist,
              auxXlist = CCLEdataset$auxXlist,
              B=NULL,
              L=NULL,eta=1,lamseq = lamLassoseq,
              tol=1e-04,maxiter=100) -> PooledNRres

  cv.twostep(Y=CCLEdataset_trainingtest[[i]]$Ytraining,
             X=CCLEdataset_trainingtest[[i]]$Xtraining,
             B=NULL, L=NULL,eta=1,
             lamseq_w = lamseq_w,
             lamseq_delta=lam_delta,
             tol=1e-04,maxiter=100,
             auxYlist=CCLEdataset$auxYlist,
             auxXlist=CCLEdataset$auxXlist) -> TwosteptransRes

  MSDtrans(Y=CCLEdataset_trainingtest[[i]]$Ytraining,
           X=CCLEdataset_trainingtest[[i]]$Xtraining,
           B=NULL, L=NULL,eta=1,
           lamseq_w = lamseq_w,
           lamseq_delta=lam_delta,
           tol=1e-04,maxiter=100,
           auxYlist=CCLEdataset$auxYlist,
           auxXlist=CCLEdataset$auxXlist,
           nfold=5,nfold_choiceforC=3,
           nfold_selectionstep_pe=3,
           C=c(0.001, 0.01,  0.05,  0.1)) -> MSDRes

  FSDtrans(Y=CCLEdataset_trainingtest[[i]]$Ytraining,
           X=CCLEdataset_trainingtest[[i]]$Xtraining,
           B=NULL, L=NULL,eta=1,
           lamseq_w = lamseq_w,
           lamseq_delta=lam_delta,
           tol=1e-04,maxiter=100,
           auxYlist=CCLEdataset$auxYlist,
           auxXlist=CCLEdataset$auxXlist,
           nfold=5,nfold_choiceforC=3,
           nfold_selectionstep_pe=3,
           C=c(0.001, 0.01,  0.05,  0.1)) -> FSDRes


  predLasso <- cbind(1,CCLEdataset_trainingtest[[i]]$Xtest) %*% cvLasso$B
  Lassoerr <- mean( (predLasso-CCLEdataset_trainingtest[[i]]$Ytest)^2 ,na.rm=TRUE)

  predPooledNR <- cbind(1,CCLEdataset_trainingtest[[i]]$Xtest) %*% PooledNRres$B
  PooledNRerr <- mean( (predPooledNR-CCLEdataset_trainingtest[[i]]$Ytest)^2 ,na.rm=TRUE)

  predTwostep <- cbind(1,CCLEdataset_trainingtest[[i]]$Xtest) %*% TwosteptransRes$B
  Twosteperr <- mean( (predTwostep-CCLEdataset_trainingtest[[i]]$Ytest)^2 ,na.rm=TRUE)

  predMSD<- cbind(1,CCLEdataset_trainingtest[[i]]$Xtest) %*% MSDRes$B
  MSDerr <- mean( (predMSD-CCLEdataset_trainingtest[[i]]$Ytest)^2 ,na.rm=TRUE)

  predFSD <- cbind(1,CCLEdataset_trainingtest[[i]]$Xtest) %*% FSDRes$B
  FSDerr <- mean( (predFSD-CCLEdataset_trainingtest[[i]]$Ytest)^2 ,na.rm=TRUE)



  sourceindex <- FSDRes$detectedsources
  if(length(sourceindex)==0){
    FSDSCADerr <- Lassoerr
    Bscad <- cvLasso$B
    rankSCAD <- sum(svd(matrix(Bscad[-1,],p,q))$d > 1e-02)
  }else{
    auxYlist_SCAD <- auxXlist_SCAD <- list()
    for(ss in 1:length(sourceindex)){
      auxYlist_SCAD[[ss]] <- CCLEdataset$auxYlist[[sourceindex[ss]]]
      auxXlist_SCAD[[ss]] <- CCLEdataset$auxXlist[[sourceindex[ss]]]
    }
    FSDSCADres <- BIC.TransSCAD(Y=CCLEdataset_trainingtest[[1]]$Ytraining,
                                X=CCLEdataset_trainingtest[[1]]$Xtraining,
                                auxYlist=auxYlist_SCAD,
                                auxXlist=auxXlist_SCAD,
                                lamseq_w=lamseq_w_SCAD,
                                lamseq_delta=lamseq_delta_SCAD,
                                eta=1, a=3.7,
                                B=NULL,L=NULL,Delta=NULL,H=NULL,Pi=NULL,
                                maxiter_inital=300, maxiter_biascorrection=300,
                                tol_inital=1e-04, tol_biascorrection=1e-04,
                                standardize=T)
    predFSDSCAD <- cbind(1,CCLEdataset_trainingtest[[i]]$Xtest) %*% FSDSCADres$B
    FSDSCADerr <- mean( (predFSDSCAD-CCLEdataset_trainingtest[[i]]$Ytest)^2 ,na.rm=TRUE) # MSPE
    Bscad <- FSDSCADres$B
  }


  error_i <- c(Lassoerr, PooledNRerr, Twosteperr, MSDerr, FSDerr, FSDSCADerr)
  names(error_i) <- c("NR", "Pooled-NR", "[K]-Trans", "MSD-Trans", "FSD-Trans", "FSD-Trans-SCAD")
  errorlist[[i]] <- error_i # save test errors

  Blist_i <- list(cvLasso$B, PooledNRres$B, TwosteptransRes$B,
                  MSDRes$B, FSDRes$B, Bscad)

  names(Blist_i) <- c("NR", "Pooled-NR", "[K]-Trans", "MSD-Trans", "FSD-Trans", "FSD-Trans-SCAD")
  Blist[[i]]  <- Blist_i # save Bhat

  ranklist_i <- unlist(lapply(Blist_i, function(x){
    sum(svd(matrix(x[-1,],p,q))$d > 1e-02)
  }))
  names(ranklist_i) <- c("NR", "Pooled-NR", "[K]-Trans", "MSD-Trans", "FSD-Trans", "FSD-Trans-SCAD")
  ranklist[[i]] <- ranklist_i



}
```

예측오차

```
apply(do.call('rbind',errorlist),2,mean)
```

```
##             NR      Pooled-NR      [K]-Trans      MSD-Trans      FSD-Trans
##      0.6018483      0.6251835      0.6201859      0.5809819      0.5657689
## FSD-Trans-SCAD
##      0.4861628
```

```
apply(do.call('rbind',errorlist),2,sd)
```

```
##             NR      Pooled-NR      [K]-Trans      MSD-Trans      FSD-Trans
##     0.13327427     0.09017468     0.16950627     0.10428314     0.13009386
## FSD-Trans-SCAD
##     0.03702648
```

```
#### mean ranks
apply(do.call('rbind',ranklist),2,mean)
```

```
##             NR      Pooled-NR      [K]-Trans      MSD-Trans      FSD-Trans
##       1.666667       4.000000       4.000000       3.333333       2.666667
## FSD-Trans-SCAD
##       1.000000
```

```
#### standard deviation of ranks
apply(do.call('rbind',ranklist),2,sd)
```

```
##             NR      Pooled-NR      [K]-Trans      MSD-Trans      FSD-Trans
##      0.5773503      0.0000000      0.0000000      1.1547005      1.1547005
## FSD-Trans-SCAD
##      0.0000000
```

Wilconxon test

FSD vs NR

```
#### Wilcoxon singed rank tests
FSDvsNR <- wilcox.test(do.call('rbind',errorlist)[,5],
            do.call('rbind',errorlist)[,1], paired = TRUE, alternative = "two.sided")
FSDvsNR$p.value
```

```
## [1] 0.25
```

FSD vs Pooled-NR

```
#### Wilcoxon singed rank tests
FSDvsNR <- wilcox.test(do.call('rbind',errorlist)[,5],
            do.call('rbind',errorlist)[,2], paired = TRUE, alternative = "two.sided")
FSDvsNR$p.value
```

```
## [1] 0.25
```

FSD vs [K]-Trans

```
#### Wilcoxon singed rank tests
FSDvsNR <- wilcox.test(do.call('rbind',errorlist)[,5],
            do.call('rbind',errorlist)[,3], paired = TRUE, alternative = "two.sided")
FSDvsNR$p.value
```

```
## [1] 0.5
```

FSD vs MSD

```
#### Wilcoxon singed rank tests
FSDvsNR <- wilcox.test(do.call('rbind',errorlist)[,5],
            do.call('rbind',errorlist)[,4], paired = TRUE, alternative = "two.sided")
```

```
## Warning in wilcox.test.default(do.call("rbind", errorlist)[, 5],
## do.call("rbind", : cannot compute exact p-value with zeroes
```

```
FSDvsNR$p.value
```

```
## [1] 1
```

FSD vs FSD-SCAD

```
#### Wilcoxon singed rank tests
FSDvsNR <- wilcox.test(do.call('rbind',errorlist)[,5],
            do.call('rbind',errorlist)[,6], paired = TRUE, alternative = "two.sided")
FSDvsNR$p.value
```

```
## [1] 0.5
```