Cross validation procedures

YipengSong

July 14, 2017

This document is used to show the results of EM-Wold cross validation procedures on a simulated binary data set. The interpretation should be refered to the paper.

1. Binary data simulation

Binary data with low dimensional structure can be simulated from logistic SVD model.

```
library(MASS)   # for mutivariate gaussian distribution
library(pracma) # for matrix computation

set.seed(123)

## parameters used in the binary simulation
m   = 99; n = 50;       # samples and variables
mu  = matrix(rep(0,n)); # offset term
C   = 10    # constant to adjust the scale of AB^T
R_real = 3  # low rank

## simulation of loading matrix B
B_raw  = matrix(rnorm(n*R_real, sd=1), nrow=n, ncol=R_real)
B_real = gramSchmidt(B_raw)$Q

## simulation of score matrix A
# specify three group means
a1=matrix(c(2,-1,3)); a2=matrix(c(-1,3,-2)); a3=matrix(c(-1,-2,-1))
A1_real = mvrnorm(m/3, mu=a1, Sigma=1*eye(R_real,R_real))
A2_real = mvrnorm(m/3, mu=a2, Sigma=1*eye(R_real,R_real))
A3_real = mvrnorm(m/3, mu=a3, Sigma=1*eye(R_real,R_real))
A_real  = rbind(A1_real,A2_real, A3_real)

## simulation of theta matrix and probability matrix
Theta = matrix(rep(1,m))%*%t(mu) + C*A_real%*%t(B_real) # logistic SVD model
Theta_prob = sigmoid(Theta) # transform to 0-1 scale probability

## generating binary data according to corresponding probability
X = matrix(data=0, nrow=m, ncol=n)
# sampling from bernoulli distribution
for(i in 1:m){
  for(j in 1:n){
    X[i,j] = rbinom(1,1,Theta_prob[i,j])
  }
}
```

2. Cross validation procedures

EM-Wold cross validation is used.

```
## EM-wold CV procedures are encapsulated into functions in "code_EM_wold_cross-validation.R"
# import CV procedures
source(file = "./Rcode/code_EM_wold_cross-validation.R")

## EM-wold CV of PCA model
CV.pca = CV_pca_wold(X, # binary data
                     7, # K-fold CV
                     2) # number of PCs
train_error = CV.pca$train_error # total error and balanced error
CV_error    = CV.pca$cv_error    # total error and balanced error
opt_thres   = CV.pca$opt_thres   # thresholds in binarizing continuous estimation
CV_error
```

```
##     total  balanced
## 0.1500962 0.1496949
```

```
opt_thres
```

```
##     total  balanced
## 0.4747475 0.4747475
```

```
## EM-wold CV of Gifi model
CV.gifi = CV_gifi_wold(X, # binary data
                       7, # K-fold CV
                       2) # number of PCs
train_error = CV.gifi$train_error # total error and balanced error
CV_error    = CV.gifi$cv_error    # total error and balanced error
opt_thres   = CV.gifi$opt_thres   # thresholds in binarizing continuous estimation
CV_error
```

```
##     total  balanced
## 0.1502983 0.1498972
```

```
opt_thres
```

```
##     total  balanced
## 0.4747475 0.4747475
```

```
## EM-wold CV of logistic PCA model
CV.logpca = CV_logpca_wold(X, # binary data
                           7, # K-fold CV
                           2) # number of PCs
train_error = CV.logpca$train_error # total error and balanced error
CV_error    = CV.logpca$cv_error    # total error and balanced error
opt_thres   = CV.logpca$opt_thres   # thresholds in binarizing continuous estimation
CV_error
```

```
##     total  balanced
## 0.1662600 0.1659734
```

```
opt_thres
```

```
##     total  balanced
## 0.4848485 0.4848485
```

```
## EM-wold CV of logistic SVD model
CV.logsvd = CV_logsvd_wold(X, # binary data
                           7, # K-fold CV
```

```
                            2) # number of PCs
train_error = CV.logsvd$train_error # total error and balanced error
CV_error    = CV.logsvd$cv_error    # total error and balanced error
opt_thres   = CV.logsvd$opt_thres   # thresholds in binarizing continuous estimation
CV_error
```

```
##     total  balanced
## 0.1597923 0.1595774
```

```
opt_thres
```

```
##     total  balanced
## 0.4949495 0.4949495
```