HW7_S610

Jongwook Kim

11/17/2019

1.Download the data from http://jfukuyama.github.io/teaching/stat610/assignments/ hw7.csv. The rows of the matrix are the samples, and the columns are variable measure- ments. You should have 50 samples and 10 variables.

```
setwd("~/Dropbox/data")
dat <- read.csv("hw7.csv")
dat <- dat[,-1] #remove X column
dat <- as.matrix(dat)
```

2.Estimate the inverse covariance matrix using the graphical lasso.

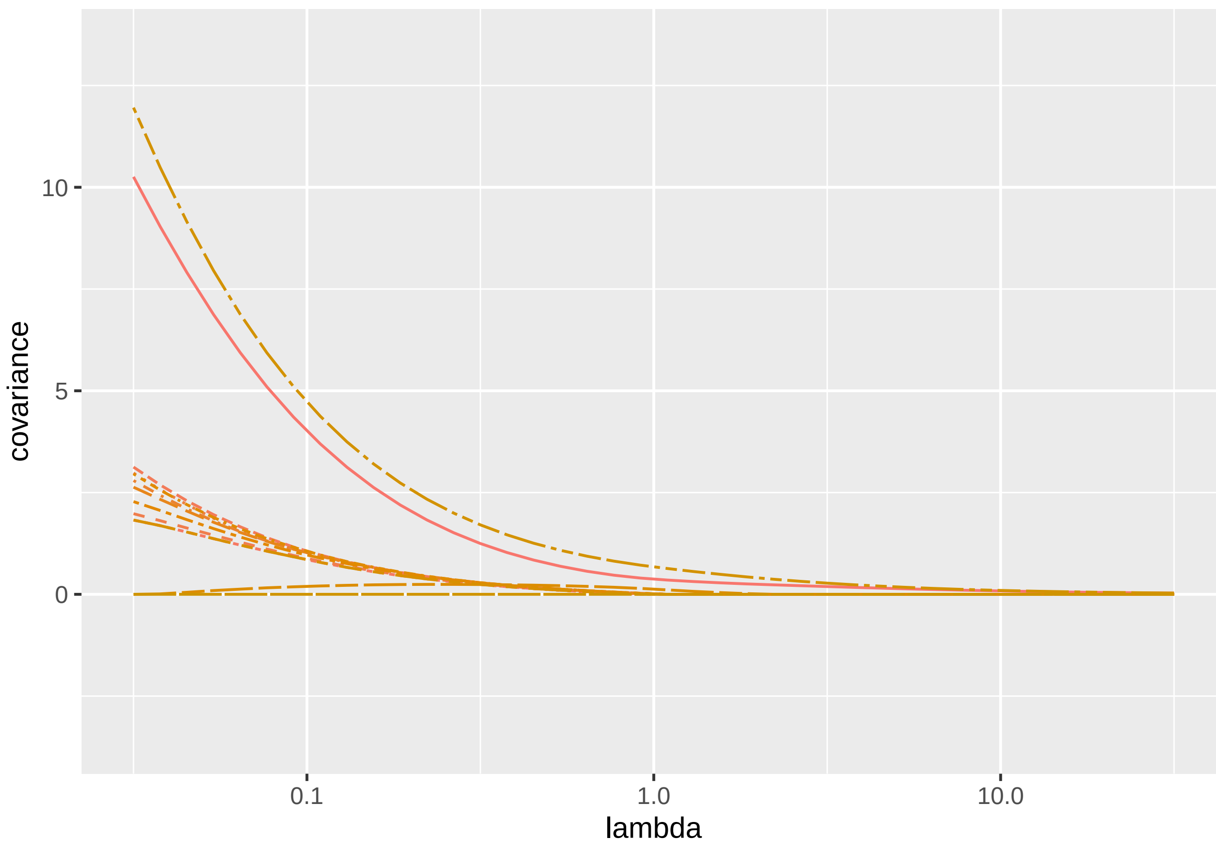
```
library(CVXR)
```

```
##
## Attaching package: 'CVXR'

## The following object is masked from 'package:stats':
##
##     power
```

```
library(ggplot2)

e <- matrix((rep(1,nrow(dat))))
px <- (diag(nrow(dat)) - e%*%t(e)/nrow(dat))%*%dat #centering dataset
S <- (t(px)%*%px)/(nrow(px)-1)

lambda_search <- 10^(seq(-1.5, 1.5, length.out = 40))

get_theta_lasso <- function(lambda){
  theta <- Variable(10,10)
  objective <- Minimize(-log_det(theta) + matrix_trace(S%*%theta) + lambda*sum(abs(theta)))
  problem <- Problem(objective)
  result <- solve(problem)
  return(result$getValue(theta))
}
#theta_hat <- plyr::alply(lambda_search, 1, get_theta_lasso)
theta_hat2 <- plyr::aaply(lambda_search, 1, get_theta_lasso)
```

3. Choose some subset of the elements of the inverse covariance and plot the estimates for a variety of values of λ.

```
covar <- theta_hat2[,1,] #make each element as column to make the plot.
for(i in 2:10){
  covar <- cbind(covar,theta_hat2[,i,])
}
covar <- cbind(lambda=lambda_search, covar)

theta_melted = reshape2::melt(data.frame(covar), id.vars = "lambda", value.name = "covariance")

#Since covariance matrix is a 10 x 10 matrix, ggplot will have 100 lines.
ggplot(theta_melted) +
    geom_line(aes(x = lambda, y = covariance, color = variable, lty = variable)) +
    scale_x_log10() + theme(legend.position = "none")
```



4. We would like to pick a good value of λ: one way to do this is by cross validation. The idea is to choose the value of λ that gives the highest value of the likelihood on a held-out portion of the data.

```
#Randomly shuffle the data
cv_dat <- dat[sample(nrow(dat)),]
```

```r
#Create 10 equally size folds
folds <- cut(seq(1,nrow(cv_dat)),breaks=10,labels=FALSE)

get_theta_lasso_cv <- function(lambda){
  nl <- 0 #the initial value of the negative log likelihood
  #Perform 10 fold cross validation
  for(i in 1:10){
    #Segement the data by fold using the which() function
    testIndexes <- which(folds==i,arr.ind=TRUE)
    test <- cv_dat[testIndexes, ]
    train <- cv_dat[-testIndexes, ]

    e <- matrix((rep(1,nrow(train))))
    px <- (diag(nrow(train)) - e%*%t(e)/nrow(train))%*%train #centering dataset
    S_train <- (t(px)%*%px)/(nrow(px)-1)

    theta <- Variable(10,10)
    objective <- Minimize(-log_det(theta) + matrix_trace(S_train%*%theta) + lambda*sum(abs(theta)))
    problem <- Problem(objective)
    result <- solve(problem)
    theta_hat <- result$getValue(theta) #estimate of corvariance matrix
    nl2 <- -log(det(theta_hat)) + sum(diag(S_train%*%theta_hat)) #negative log-likelihood
    nl <- nl + nl2
  }
  return(nl)
}

#lambda_search <- 10^(seq(-1.5, 1.5, length.out = 40))
n_like <- plyr::aaply(lambda_search, 1, get_theta_lasso_cv) #negative likelihoods with different lambda
like <- data.frame(lambda=lambda_search, negative_log_likelihood=n_like)
like[which(like$negative_log_likelihood==min(like$negative_log_likelihood)),] #lambda achieving the max
```

```
##       lambda negative_log_likelihood
## 1 0.03162278               -177.7601
```

```r
knitr::kable(t(like), align = 'c')
```

	1	2	3	4	5	6	
lambda	0.0316228	0.0377505	0.0450657	0.0537984	0.0642233	0.0766682	0.0
negative_log_likelihood	-177.7601187	-168.0368119	-157.6214336	-146.5605078	-134.9025973	-122.6978749	-10

```r
lambda <- like[which(like$negative_log_likelihood==min(like$negative_log_likelihood)),1]
result <- get_theta_lasso(lambda)
round(result,2)#theta_hat by cv
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10]
## [1,] 10.26  1.82  3.13  1.98  2.98  2.79  2.63  2.28  2.96  0.00
## [2,]  1.82 11.96  0.00  0.00  0.00  0.00 -0.08  0.00 -0.71 -3.11
## [3,]  3.13  0.00 12.47 -0.37  0.00  0.00  0.00 -0.11  0.00 -2.56
## [4,]  1.98  0.00 -0.37 11.49  0.00  0.00  0.00 -0.08  0.00 -3.18
## [5,]  2.98  0.00  0.00  0.00 12.15  0.00 -0.22 -0.12  0.00 -1.91
## [6,]  2.79  0.00  0.00  0.00  0.00 13.53 -0.63  0.00  0.00 -3.55
```

```
## [7,]  2.63 -0.08  0.00  0.00 -0.22 -0.63 12.58 -0.43  0.00 -3.05
## [8,]  2.28  0.00 -0.11 -0.08 -0.12  0.00 -0.43 11.57  0.00 -3.32
## [9,]  2.96 -0.71  0.00  0.00  0.00  0.00  0.00  0.00 12.74 -3.12
## [10,]  0.00 -3.11 -2.56 -3.18 -1.91 -3.55 -3.05 -3.32 -3.12 11.49
```

5. It turns out that in this dataset, the first variable is a measure of the abundance of a keystone predator, the second through 9th variables are measures of the abundances of prey species, and the last variable is is a measure of the abundance of a food source. Perform maximum likelihood estimation under the constraint that the partial correlations between the prey species are zero.

```
theta <- Variable(10,10)
objective <- Minimize(-log_det(theta) + matrix_trace(S%*%theta))
constraints <- list(theta[2,3:9] == 0, theta[3,4:9] == 0, theta[3,4:9] == 0, theta[4,5:9] == 0, theta[5

problem <- Problem(objective, constraints)
result2 <- solve(problem)
round(result2$getValue(theta), 2) #estimator of theta
```

```
##       [,1]   [,2]  [,3]   [,4]  [,5]   [,6]   [,7]   [,8]   [,9]  [,10]
##  [1,] 40.63   3.68 18.27   4.20 15.95  17.41  13.96   6.85  16.77  -8.14
##  [2,]  3.68  44.09  0.00   0.00  0.00   0.00   0.00   0.00   0.00 -16.50
##  [3,] 18.27   0.00 55.97   0.00  0.00   0.00   0.00   0.00   0.00  -9.23
##  [4,]  4.20   0.00  0.00  40.67  0.00   0.00   0.00   0.00   0.00 -15.31
##  [5,] 15.95   0.00  0.00   0.00 43.60   0.00   0.00   0.00   0.00  -3.03
##  [6,] 17.41   0.00  0.00   0.00  0.00  94.20   0.00   0.00   0.00 -30.24
##  [7,] 13.96   0.00  0.00   0.00  0.00   0.00  63.53   0.00   0.00 -18.95
##  [8,]  6.85   0.00  0.00   0.00  0.00   0.00   0.00  47.00   0.00 -17.90
##  [9,] 16.77   0.00  0.00   0.00  0.00   0.00   0.00   0.00  67.49 -18.11
## [10,] -8.14 -16.50 -9.23 -15.31 -3.03 -30.24 -18.95 -17.90 -18.11  56.18
```

6. Obtain bootstrap confidence intervals for the non-zero elements of θ: for some reasonably large number B, perform the following:

```
bootstrap_ci = function(data, estimator, alpha, B) {
    boot_estimates = get_boot_estimates(data, estimator, B)
    boot_ci = get_ci(boot_estimates, alpha)
    return(boot_ci)
}
get_ci = function(estimates, alpha) {
    q_lo = alpha / 2
    q_hi = 1 - (alpha / 2)
    if(!is.null(dim(estimates))) {
        ## if we have multi-dimensional estimates
        cis = plyr::adply(estimates, c(1,2), function(x) quantile(x, probs = c(q_lo, q_hi)))
    } else {
```

```r
        ## if we have one-dimensional estimates
        cis = quantile(estimates, probs = c(q_lo, q_hi))
    }
    return(cis)
}
get_boot_estimates = function(data, estimator, B) {
    boot_estimates = replicate(B, expr = {
        resampled_data = get_bootstrap_sample(data)
        boot_estimate = estimator(resampled_data)
        return(boot_estimate)
    })
    return(boot_estimates)
}
get_bootstrap_sample = function(data) {
    if(!is.null(dim(data))) {
        boot_sample = bootstrap_sample_rows(data)
    } else {
        boot_sample =  bootstrap_sample_elements(data)
    }
    return(boot_sample)
}
bootstrap_sample_rows = function(data) {
    n = nrow(data)
    boot_idx = sample(1:n, size = n, replace = TRUE)
    bootstrap_sample = data[boot_idx,]
    return(bootstrap_sample)
}
bootstrap_sample_elements = function(data) {
    n = length(data)
    boot_idx = sample(1:n, size = n, replace = TRUE)
    bootstrap_sample = data[boot_idx]
    return(bootstrap_sample)
}
estimator <- function(data){ #estimator fn for the covariance matrix(theta)
  e <- matrix((rep(1,nrow(data))))
  px <- (diag(nrow(data)) - e%*%t(e)/nrow(dat))%*%data #centering dataset
  S <- (t(px)%*%px)/(nrow(px)-1)

  theta <- Variable(10,10)
  objective <- Minimize(-log_det(theta) + matrix_trace(S%*%theta))
  constraints <- list(theta[2,3:9] == 0, theta[3,4:9] == 0, theta[3,4:9] == 0, theta[4,5:9] == 0, theta

  problem <- Problem(objective, constraints)
  result2 <- solve(problem)
  theta_hat <- result2$getValue(theta) #estimator of theta
  #theta_ij <- matrix(theta_hat[which(round(theta_hat,5) != 0)],ncol=1) #extract only elements not zero
  #return(theta_ij)
  return(theta_hat)
}

CI <- bootstrap_ci(data = dat, estimator = estimator, alpha = .05, B = 20)
CI$X1 <- as.numeric(CI$X1); CI$X2 <- as.numeric(CI$X2);
CI <- round(CI,4)
```

```r
CI[which(CI[,3] != 0),]
```

```
##      X1 X2     2.5%    97.5%
## 1     1  1  37.6049  63.5476
## 2     2  1  -7.5323  11.1167
## 3     3  1   9.6818  35.9784
## 4     4  1  -0.3103  14.3620
## 5     5  1  13.6247  24.7547
## 6     6  1   5.9611  26.5649
## 7     7  1   7.2725  23.1017
## 8     8  1  -1.2585  13.5734
## 9     9  1   8.4942  28.6552
## 10   10  1 -13.9294   0.7973
## 11    1  2  -7.5323  11.1167
## 12    2  2  36.1948  66.1218
## 20   10  2 -30.0618 -10.5116
## 21    1  3   9.6818  35.9784
## 23    3  3  43.1854  96.1178
## 30   10  3 -22.3516  -0.9856
## 31    1  4  -0.3103  14.3620
## 34    4  4  34.1478  61.8097
## 40   10  4 -22.4010  -8.8997
## 41    1  5  13.6247  24.7547
## 45    5  5  37.2701  75.1413
## 50   10  5 -10.1486   2.6867
## 51    1  6   5.9611  26.5649
## 56    6  6  68.2771 153.9886
## 60   10  6 -57.4199 -20.5968
## 61    1  7   7.2725  23.1017
## 67    7  7  47.3315  92.1603
## 70   10  7 -35.5361 -12.4006
## 71    1  8  -1.2586  13.5734
## 78    8  8  45.3587  64.5063
## 80   10  8 -30.1467 -12.7091
## 81    1  9   8.4942  28.6552
## 89    9  9  54.1760  90.9199
## 90   10  9 -23.9778  -8.9539
## 91    1 10 -13.9294   0.7973
## 92    2 10 -30.0618 -10.5116
## 93    3 10 -22.3516  -0.9856
## 94    4 10 -22.4010  -8.8997
## 95    5 10 -10.1486   2.6867
## 96    6 10 -57.4199 -20.5968
## 97    7 10 -35.5361 -12.4006
## 98    8 10 -30.1467 -12.7091
## 99    9 10 -23.9778  -8.9539
## 100  10 10  54.4601  81.3310
```