A User's guide for package CAPIT

Mengjie Chen

August 2016

1 Introduction

This guide provides a tour of package `CAPIT`, which inplements Sparse CCA via Precision Adjusted Iterative Thresholding (CAPIT) procedure proposed in Chen et al. This manual is composed of test-runs on basic functions with simulated datasets. The main purpose is to reproduce the results shown in simulation studies.

2 A probabilistic model of sparse CCA

Canonical correlation analysis (CCA) is a celebrated technique proposed by Hotelling to find the linear combinations of two sets of random variables with maximal correlation. Taking an example from integrative cancer genomics studies, CCA can be used to study the relationship between DNA methylation and gene expression. Let $X \in \mathbb{R}^{p_1}$ be a centered random vector representing p_1 methylation probes and $Y \in \mathbb{R}^{p_2}$ be a centered random vector representing p_2 genes. The correlation between DNA methylation and gene expression is defined as

$$\max_{(a,b)} \left\{ \mathrm{Cov}(a^T X, b^T Y) : \mathrm{Var}(a^T X) = \mathrm{Var}(b^T Y) = 1 \right\}. \tag{1}$$

The maximizer (θ, η) is the linear combinations of X and Y that are maximally correlated. They are termed as the canonical directions. Under high-dimensional settings where p_1 and p_2 are large, the canonical directions (θ, η) can be sparse and only small subsets of methylation probes and genes are significantly correlated. This gives rise to the sparse CCA problem to be studied in this paper, the aim of which is to find sparse canonical directions between two large sets of random variables.

To study CCA from a theoretical point of view, we propose a probabilistic model that naturally characterizes the optimization procedure. Given the covariance structure, we present the following proposition.

Proposition 2.1 *When Σ_{12} is of rank 1, the solution (up to sign jointly) of is (θ, η) if and only if the cross covariance between X and Y can be written as*

$$\Sigma_{12} = \lambda \Sigma_1 \theta \eta^T \Sigma_2,$$

where $0 < \lambda \leq 1$, $\theta^T \Sigma_1 \theta = 1$ and $\eta^T \Sigma_2 \eta = 1$. As a consequence, the correlation between $a^T X$ and $b^T Y$ are maximized by $\mathrm{corr}(\theta^T X, \eta^T Y)$, and λ is the canonical correlation between X and Y.

3 CAPIT algorithm

The CAPIT algorithm has two steps, first step is to estimate covariance/precison matrix and the second step is iterative thresholding. For the first step, we implement three approaches for sparse covariance matrix estimation, hard/soft thresholding, tapering and toeplitz. We also allow direct input of precision matrix from other methods. For the second step, we implement an iterative thresholding algorithm, which selects thresholding levels through cross validation. Pre-specified thresholding levels are also allowed.

4 Test run on `CAPIT()` on simulated data

We simulate a scenario when the covariance matrices Σ_1 and Σ_2 are sparse. More specifically, the covariance matrix $\Sigma_1 = \Sigma_2 = (\sigma_{ij})_{1 \leq i,j \leq p}$ takes the form

$$\sigma_{ij} = \rho^{|i-j|} \quad \text{with} \quad \rho = 0.2.$$

The canonical pair (θ, η) is generated by normalizing a vector taking the same value at the coordinates $(1, 6, 11, 16, 21)$ and zero elsewhere such that $\theta^T \Sigma_1 \theta = 1$ and $\eta^T \Sigma_2 \eta = 1$. The canonical correlation λ is set as 0.9. We generate the $2n \times p$ data matrices X and Y jointly from the model in the proposition.

```
> n <- 750*2
> p1 <- 200
> p2 <- 200
> s1 <- 5
> s2 <- 5
> rho <- 0.2
> Sigma1 <- matrix(0, ncol = p1, nrow = p1)
> for(i in 1:p1){
+         for(j in 1:p1){
```

```
+                 Sigma1[i, j] <- rho^(abs(i - j))
+         }
+ }
> Sigma2 <- matrix(0, ncol = p2, nrow = p2)
> for(i in 1:p2){
+         for(j in 1:p2){
+                 Sigma2[i, j] <- rho^(abs(i - j))
+         }
+ }
> theta <- as.matrix(c(rep(c(1, 0, 0, 0, 0), s1), rep(0, p1-5*s1)))
> theta <- theta/as.numeric(sqrt(t(theta)%*%Sigma1%*%theta))
> eta <- as.matrix(c(rep(c(1, 0, 0, 0, 0), s2), rep(0, p2-5*s2)))
> eta <- eta/as.numeric(sqrt(t(eta)%*%Sigma2%*%eta))
> lambda <- 0.9
> sigma_cov <- rbind(cbind(Sigma1, lambda*Sigma1%*%theta%*%t(eta)%*%Sigma2),
+                    cbind(lambda*Sigma2%*%eta%*%t(theta)%*%Sigma1, Sigma2))
> require(MASS)
> set.seed(100)
> Z <- mvrnorm(n, rep(0, p1+p2), sigma_cov)
> X <- Z[, 1:p1]
> Y <- Z[, (p1+1):(p1+p2)]
> v <- c(theta, eta)
```

The inputs of function `scPLS` include two matrices X and Y. We split the data into two halves. We use the first half to estimate sparse covariance matrix. The default method to estimate sparse covariance matrix is hard thresholding. If set `selection="soft"`, then soft thresholding will be used instead. To select the thresholding level, we further split the first part of the data into a $2:1$ training set and tuning set. We select the tuning parameter by minimizing the distance of the estimated covariance from the training set and sample covariance matrix of the tuning set in term of the Frobenius norm. The tuning parameter is selected through a screening on 50 numbers in the interval of $[0.01, 0.5]$.

The levels for iterative thresholding are obtained through cross validation. Specifically, we split the second part of the data into a $2:1$ training set and tuning set. We use the following natural levels: $t_{ij} = c_t(\|\hat{\Omega}_1\| + \|\hat{\Omega}_2\|)$ and $\gamma_i = c_g(\|\hat{\Omega}_1\| + \|\hat{\Omega}_2\|)\sqrt{\frac{\log p}{n}}$. The constants c_t and c_g are selected by maximizing $\hat{\theta}^T\hat{\Sigma}_{12}^{tra}\hat{\eta}/(\hat{\theta}^T\hat{\Sigma}_1^{tra}\hat{\theta}\hat{\eta}^T\hat{\Sigma}_2^{tra}\hat{\eta})^{1/2}$, where $\hat{\Sigma}_{12}^{tra}, \hat{\Sigma}_1^{tra}, \hat{\Sigma}_2^{tra}$ are the sample cross covariance between X and Y, sample covariance of X and sample covariance of Y respectively using the tuning set while $(\hat{\theta}, \hat{\eta})$ is the estimated canonical direction pair using the training set. Specifically, both c_t and c_g are selected through a

screening on 7 numbers in the interval of $[0.5, 3]$. This interval can be changed through modifying argument `search.grid`.

```
> library(CAPIT)
> u1 <- CAPIT(X, Y)
```

The output of `CAPIT` is a list, which further contains two list `res` and `resOLS`. `res` contains `alpha` and `beta`, which returns canonical vectors of X and Y estimated from the vanilla CAPIT algorithm, respectively. `res` contains `alpha` and `beta`, which returns canonical vectors of X and Y estimated from CAPIT algorithm with refinement using ordinary least squares, respectively. We can check the estimation errors for (θ, η) as measured by $L(\hat{\theta}, \theta) \vee L(\hat{\eta}, \eta)$. Here we use Frobenius loss.

```
> FLoss(unlist(u1[[1]]), v, p1) #CAPIT

[1] 0.08533437

> FLoss(unlist(u1[[2]]), v, p1) #CAPIT + refinement by OLS

[1] 0.04722038
```

Similarly, we can estimate sparse covariance matrix using "Tapering" method or "Toeplitz" method. Both methods make stronger assumptions about the structure of covariance matrix. If the"Tapering" method is used, the elements in the covariance matrix are assumed to decay as they move away from the diagonal. For the tapering procedure, we implemented the method proposed in Cai et al(2013). If the "Toeplitz" method is used, the Toeplitz structure of the covariance matrix is assumed. We implemented the method proposed in Cai et al(2012) to estimate Toeplitz covariance matrix. Same as described previously, to select the tuning parameters, we further split the first part of the data into a $2:1$ training set and tuning set. We select the tuning parameter by minimizing the distance of the estimated covariance from the training set and sample covariance matrix of the tuning set in term of the Frobenius norm. More specifically, the bandwidths in the Toeplitz method and in the tapering method are selected through a screening on numbers in the interval of $(1, p)$. The decay rate in "tapering" algorithm `alpha` is set to be 0.3 by default.

```
> u2 <- CAPIT(X, Y, method = "toeplitz")
> FLoss(unlist(u2[[1]]), v, p1) #CAPIT

[1] 0.09608247
```

```
> FLoss(unlist(u2[[2]]), v, p1) #CAPIT + refinement by OLS

[1] 0.04706011

>

> u3 <- CAPIT(X, Y, method = "tapering")
> FLoss(unlist(u3[[1]]), v, p1) #CAPIT

[1] 0.1191908

> FLoss(unlist(u3[[2]]), v, p1) #CAPIT + refinement by OLS

[1] 0.04775202
```