Using the hdInference package

Renxiong Liu and Yunzhang Zhu

2021-08-15

The `hdInference` package contains the implementations of likelihood ratio test for a subset of parameters in Gaussian graphical model. The testing problem can be formalized as:

$$\hat{\theta}^{(0)} = \arg\max_{\theta} \ L_n(\theta) \quad \text{subject to:} \sum_{i \notin B} p_\tau(|\theta_i|) \leq K \text{ and } \theta_B = 0$$
$$\hat{\theta}^{(1)} = \arg\max_{\theta} \ L_n(\theta) \quad \text{subject to:} \sum_{i \notin B} p_\tau(|\theta_i|) \leq K,$$

where $L_n(\theta) = \sum_{i=1}^n \log p_\theta(X_i)$ is the log-likelihood for Gaussian graphical model, $p_\tau(x) = \min(x/\tau, 1)$ is the truncated L_1 function as the surrogate function of L_0 function, and (K, τ) are nonnegative tuning parameters. The details of proposal can be found in Zhu, Y., Shen, X., Pan, W. (2020). *On High-Dimensional Constrained Maximum Likelihood Inference*.

Using `glasso_nonconvex_constrained_cv` function

To use the `glasso_nonconvex_constrained_cv` function in our `hdInference` package, you need to specify

1. **sim**: a list containing a $n \times d$ data matrix;
2. **bound**: a vector of upper bounds for the constraint (K in above testing problem);
3. **tau**: tuning parameter for the truncated L_1 penalty (τ in above testing problem);
4. **num.fold**: fold number for cross validation.

We next elaborate the details of applying `glasso_nonconvex_constrained_cv` function with a real data example.

Real data example

We consider the ADNI-1 baseline data (adni.loni.usc.edu) for brain network analysis in our real example. To load ADNI-1 baseline data, type in R console

```
library(hdInference);
path_to_data=system.file("extdata", "ADNILongiBaseLine2.csv", package = "hdInference");
```

After loading the data, we extracted the cortical thicknesses for $p = 68$ regions of interest (ROIs) and focus on 5 features in our real data example.

```
ROI.index = 2:69
feature.index = 70:74
sr_data = read.table(file=path_to_data,header=TRUE,sep=",")
n = dim(sr_data)[1]
p = length(ROI.index)
condition = c("LMCI","AD","CN")

res.mat = matrix(0,n,length(ROI.index))
```

```r
for (i in ROI.index)
{
  data.tmp = sr_data[,c(i,feature.index)]
  tmp.fit = lm(data.tmp)
  res.mat[,i-1] = tmp.fit$residuals
}

sim = list(data = res.mat,sigmahat = cor(res.mat))
bound = p*c(9,10,11)
cv.score = glasso_nonconvex_constrained_cv(sim,bound)
#> CV fold:  1
#> CV fold:  2
#> CV fold:  3
#> CV fold:  4
#> CV fold:  5
bound.best.all = bound[which.min(apply(cv.score,2,mean))]
bound.best.all
#> [1] 612
```