Network models from observational omics data – practical

Wessel N. van Wieringen1,2

June 28, 2022

| 1Dept. of Epidemiology & Data Science
| Amsterdam Public Health research institute
| Amsterdam University Medical Centers, location VUmc

| 2Dept. of Mathematics
| Vrije Universiteit Amsterdam

| w.vanwieringen@amsterdamumc.nl

0. Preliminaries

This practical aims to acquaint the reader with the R package `porridge` (van Wieringen and Aflakparast (2022)), which implements the methodology described in W. N. van Wieringen (2019) and Wessel N. van Wieringen and Chen (2021) and more for network reconstruction from observational data.

The `porridge`-package augments the `rags2ridges`-package (Peeters, Bilgrau, and van Wieringen (2022)) with functionality for Gaussian graphical model estimation. Being its sibling the `porridge`-package partially follows `rags2ridges`-package in the function names.

The R code included builds up: it assumes code of the previous code blocks have been executed without error.

1. Generalized ridge estimation of the Gaussian graphical model

The reader is assumed to be familiar with the Gaussian graphical model, and with its ridge estimator.

Activate required libraries:

```
# load libraries
library(rags2ridges)
library(porridge)
```

We revisit the Alzheimer metabolite study described and analyzed in detail before the lunch break.

```
# load, extract and scale data for AD Class 2
data("ADdata")
ADclass1 <- scale(t(ADmetabolites[,sampleInfo$ApoEClass=="Class 1"]))
ADclass2 <- scale(t(ADmetabolites[,sampleInfo$ApoEClass=="Class 2"]))
```

The objects `ADclass1` and `ADclass2` contain the data in matrix format. Its rows and columns correspond to the samples and molecular entities (metabolites), respectively.

Create the target from the AD class 2 metabolites data using the machinery implemented in the `rags2ridges`-package and illustrated and practiced before the break.

```
# precision matrix estimation with LOOCV
Phat2 <- optPenalty.kCVauto(ADclass2,
                            lambdaMin =0.000001,
```

```
                              lambdaMax =10000,
                              lambdaInit=1)$optPrec
```

In addition to a target, we need an indicator variable for the compound types:

```
# make group indicator
groups <- colnames(ADclass1)
groups[grep("Amine",     colnames(ADclass1))] <- 1
groups[grep("Org.Acid",  colnames(ADclass1))] <- 2
groups[grep("Lip",       colnames(ADclass1))] <- 3
groups[grep("Ox.Stress", colnames(ADclass1))] <- 4
groups <- as.integer(groups)
```

With a target and a group indicator at hand, we now embork on the search for the penalty parameters. Here we look for four penalty parameters. To this end the `porridge`-package offers a K-fold cross-validation procedure.

```
# optLambdas <- optPenaltyPgen.kCVauto.groups(ADclass1,
#                               lambdaMin =rep(0.00001, 4),
#                               lambdaMax =rep(10000,   4),
#                               lambdaInit=rep(1,       4),
#                               target    =Phat2,
#                               fold      =5,
#                               groups    =groups,
#                               penalize.diag=FALSE)

# spoiler
optLambdas <- c(1.709453, 4.494467, 22.186967, 12.348735)
```

The cross-validation procedure is terribly slow, due to the re-evaluation of the estimator for each fold but also to the fact that we now search in a 4-dimensional space. Some speed is gained by setting the fold to K=5 instead of a leave-one-out procedure. Still, it takes ages. Try at home. Here, to speed up matters, the code above simply provides the optimal choice.

Note that the `optPenaltyPgen.kCVauto.groups`-function has an `penalize.diag`-option. Setting `penalize.diag=FALSE` leaves the diagonal elements of the precision matrix unpenalized as one may not want to shrink the reciprocal of marginal variances. In the `ridgeP`-funtion, this is partially achieved by the use of a suitable target as provided by the `default.target`-function.

Should one wish to use a zero target the `target`-option in the `optPenaltyPgen.kCVauto.groups`-function is to be set to `target = matrix(0, ncol(ADclass1), nrow(ADclass1))`. Again, as the execution of the `optPenaltyPgen.kCVauto.groups`-function is horrifically slow, we provide the optimal choice of the penalty parameters:

```
# spoiler
# optLambdas <- c(11.9672163, 12.6888013, 0.2881986, 6.4312858)
```

To get an impression of the structure of the penalty matrix, let us construct and plot it.

```
# construct generalized penalty matrix
lambdaMat <- c(rep(optLambdas[1], sum(groups==1)),
               rep(optLambdas[2], sum(groups==2)),
               rep(optLambdas[3], sum(groups==3)),
               rep(optLambdas[4], sum(groups==4)))
lambdaMat <- matrix(lambdaMat,
                    ncol=length(groups),
                    nrow=length(groups))
lambdaMat <- (lambdaMat + t(lambdaMat))/2
```

```
diag(lambdaMat) <- 0

# plot generalized penalty matrix
edgeHeat(lambdaMat)
```

Pay attention to the penalization of the elements of the precision matrix corresponding to conditional covariance between across compound type variates: those are penalized by the sum of the compound type-specific penalty parameters.

We now estimate the AD class 1 precision matrix.

```
# fit precision matrix
Phat1 <- ridgePgen(covML(ADclass1),
                   lambda       =lambdaMat,
                   target       =Phat2)
colnames(Phat1) <- rownames(Phat1)    <-
                   colnames(ADclass1)
```

As before, should one have chosen a zero target the `target`-option in the `ridgePgen`-function is to be set to `target = matrix(0, ncol(ADclass1), nrow(ADclass1))`.

We sparsify the obtained precision matrix as before using the `sparsify`-function.

```
# extract the CIG
Phat1  <- sparsify(Phat1,
                   threshold="localFDR",
                   FDRcut   =.999)
```

This provides the conditional independence graph of the AD class 1 metabolites, which we plot by means of the `Ugraph`-function:

```
# visualize the CIG
PcorP1  <- pruneMatrix(Phat1$sparseParCor)
Colors <- rownames(PcorP1)
Colors[grep("Amine",     rownames(PcorP1))] <- "lightblue"
Colors[grep("Org.Acid",  rownames(PcorP1))] <- "orange"
Colors[grep("Lip",       rownames(PcorP1))] <- "yellow"
Colors[grep("Ox.Stress", rownames(PcorP1))] <- "purple"

# plot network
Ugraph(PcorP1,
       type  ="fancy",
       lay   ="layout_with_fr",
       Vcolor=Colors,
       Vsize =7,
       Vcex  =0.3)
```

At first glance the obtained AD class 1 conditional independence graph resembles that of the AD class 2.

We investigate the differences between the two networks more precisely by means of their difference graph as generated by the `DiffGraph`-function. Prior to drawing this graph we sparsify the AD class 2 precision estimate.

```
# sparsify the AD class 2 precision
Phat2  <- sparsify(Phat2,
                   threshold="localFDR",
                   FDRcut   =.999)
```

```
#
Phat1 <- Phat1$sparseParCor
Phat2 <- Phat2$sparseParCor
idRemove <- which(rowSums(adjacentMat(Phat1) * adjacentMat(Phat2) -
                  adjacentMat(Phat2)) == 0)

# redefine colors
Colors <- rownames(Phat1)
Colors[grep("Amine",     rownames(Phat1))] <- "lightblue"
Colors[grep("Org.Acid",  rownames(Phat1))] <- "orange"
Colors[grep("Lip",       rownames(Phat1))] <- "yellow"
Colors[grep("Ox.Stress", rownames(Phat1))] <- "purple"

# plot differential CIG
DiffGraph(Phat1[-idRemove, -idRemove],
          Phat2[-idRemove, -idRemove],
          lay   ="layout_with_fr",
          Vcolor= Colors[-idRemove],
          Vsize = 7,
          Vcex  = .3,
          main  = "Differential Network")
```

The resulting difference graph of the AD class 1 and 2 conditional independence graph suggests a different metabolite-interaction structure among the blue nodes, i.e. the amine compound type.

Author contribution

All text and code written by Wessel N. van Wieringen.

License

References

Peeters, C F W, A E Bilgrau, and W N van Wieringen. 2022. "rags2ridges: ridge estimation of precision matrices from high-dimensional data. R package version 2.2.5," https://CRAN.R–project.org/package=rags2ridges.

van Wieringen, W. N., and M. Aflakparast. 2022. *Porridge: Ridge-Type Estimation of a Potpourri of Models.*

Wieringen, W N van. 2019. "The Generalized Ridge Estimator of the Inverse Covariance Matrix." *Journal of Computational and Graphical Statistics* 28 (4): 932–42.

Wieringen, Wessel N van, and Yao Chen. 2021. "Penalized Estimation of the Gaussian Graphical Model from Data with Replicates." *Statistics in Medicine* 40 (19): 4279–93.