Tutorial: sparse Gaussian Graphical Models

MSc in Statistics for Smart Data – Introduction to graph analysis and modeling

Julien Chiquet, November the 28, 2017

Preliminaries

Instructions. Each student *must* send an `R markdown` report generated via `R studio` to julien.chiquet@inra.fr at the end of the tutorial. This report should answer the questions by commentaries and codes generating appropriate graphical outputs. A cheat sheet of the markdown syntax can be found here.

Required packages. Check that the following packages are correctly available on your plateform:

```
library(huge)
library(igraph)
library(sand)
library(ggplot2)
library(reshape2)
```

You also need `Rstudio`, LaTeX and packages for markdown:

```
library(knitr)
library(rmarkdown)
```

1 Introduction

This practical aims to provide a quick overview of sparse Gaussian Graphical Models (GGM) and their use in the context of network reconstruction with the example fo gene interaction networks.

To this end, we rely on the `R`-package **huge**, which implements some of the most popular sparse GGM methods and provides a set of basic tools for their handling and their analysis.

The first part focuses on an empirical analysis of the statistical models used for network reconstruction. The objective is to quickly study the range of applicability of these methods. It should also give you some insights about their limitations, especially toward the interpretability of the inferred network.

The second part applies these methods to a transcriptomic data set associated to a small regulatory network for which ground truth is partially known by the biologists.

2 First part: empirical study of sparse GGM

2.1 Synthetic data generation

Multivariate Gaussian sample. Write a function `rmvnorm(n,mu,Sigma)` that draws n samples of a Gaussian vector with parameters μ and Σ and sends back a $n \times p$ matrix $\mathbf{X}$. To this end, remark that $\text{var}(Z\mathbf{\Sigma}^{1/2})$ where $Z \sim \mathcal{N}(\mathbf{0}_p, \mathbf{I}_p)$ has the same covariance as X.

From adjacency to precision matrices. Write a function `getPrecision(A)` that takes as an argument a binary symmetric adjacency matrix and computes a symmetric positive-definite matrix $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$ with the same sparsity pattern. You can use the property of positive-definiteness own by diagonal dominant matrices and draw Ω such that

$$\mathbf{\Omega} = \mathbf{A} \times c_1 + \mathbf{I}_p \times (c_2 + |\min(\text{eig}(\mathbf{A}))|),$$

where c_1, c_2 are some small constants.

From adjacency matrices to multivariate Gaussian samples. By means of the two previous questions, write a function `rggm(n,G)` that returns a matrix of Gaussian data where `G` is an `igraph` object.

2.2 Network inference

Now, to the network reconstruction at last ! Load the **huge** package. Have a quick glance at the help. The function `huge` selects the most significant partial correlation between variables, by adjusting a sparse GGM to the data. The final number of interactions (i.e., the number of edge in the reconstructed network) is controlled by a tuning parameter, the choice of which can be obtained by cross-validation.

2.2.1 Routine check

There are three variants of the function `huge` that we studied during the course: Correlation Threshold, Neighborhood-Selection and Graphical-Lasso, corresponding to the option `ct`, `mb` and `glasso`. Apply these variants to some Gaussian data generated with your function `rggm`. Use the dedicated plot function for analyzing the ouputs.

2.2.2 Network reconstruction accuracy

ROC curve. To assess the performance of the method, we use ROC curves and area under the ROC curve (AUC). It basically measures the proportion of true positve rate (here correctly recovered edges in the network) vs. false positive rate. I give you two functions to compute these quantities:

```
perf.roc <- function(theta.hat, theta.star) {

  roc <- function(theta) {
    nzero <- which(theta != 0)
    zero  <- which(theta == 0)

    true.nzero <- which(theta.star != 0)
    true.zero  <- which(theta.star == 0)

    TP <- sum(nzero %in% true.nzero)
    TN <- sum(zero %in%  true.zero)
    FP <- sum(nzero %in% true.zero)
    FN <- sum(zero %in%  true.nzero)

    recall    <- TP/(TP+FN) ## also recall and sensitivity
    fallout   <- FP/(FP+TN) ## also 1 - specificit

    res <-  round(c(fallout,recall),3)
    res[is.nan(res)] <- 0
    names(res) <- c("fallout","recall")
    return(res)
  }

  if (is.list(theta.hat)) {
    return(as.data.frame(do.call(rbind, lapply(theta.hat, roc))))
  } else {
    return(roc(theta.hat))
  }
}

perf.auc <- function(roc) {
  fallout <- c(0,roc$fallout,1)
  recall  <- c(0,roc$recall, 1)
  dx <- diff(fallout)
  return(sum(c(recall[-1]*dx, recall[-length(recall)]*dx))/2)
}
```

Numerical Experiments. We want to study the effect of the sample size on the accuracy of the three methods. Complete the following function `one.simu` that performs a simulation by computing the area under ROC curve for the `mb`, `glasso` and `correlation` approaches from a data set generated with your function. We

fix the number of node to 20, and vary the sample size from 5 to 500.

```
require(reshape2)
one.simu <- function(i) {
  cat("n =")
  d <- 25; seq.n <- c(5,15,30,50,100,250,500)
  out <- data.frame(t(sapply(seq.n, function(n) {
      cat("",n)
      # complete this
      return(setNames(c(res.glasso,res.mb,res.corthr,n,i),
                      c("glasso","correlation","sample size", "simu")))
    })))
  return(melt(out, measure.vars = 1:2, value.name = "score"))
}
```

- Use this function to perform one single simulation and represent the evolution of the AUC for each method as a function of the sample size.
- Perform a bunch (say 30) of simulations and represent the boxplots of the AUC for each method and as a function of the sample size. You may use the `parallel` package with its function `mclapply`. (or `doMC` and `foreach` if you work with Windows). This might take some time, so check your code and be patient!

3 Second part: application to E. coli regulatory network

Consider the network and expression data found in the `Ecoli.data` dataset from the **sand** package. Symmetrize the network and remove the isolated nodes. Then, infer the network from the expression data. Compare the inferred network to the reference network. You can plot the ROC curve. Finally, run the SBM from your previous tutorial to analysis the structure of the inferred network.