4850assignment2

Yao Li

20/03/2020

1 Question 1 Wine Data

1.1 (a) Nominal logistic regression

```
library(nnet)
wine = read.table("wine.txt", header = TRUE)
data = wine[,2:15]

nlr_mod = multinom(Class ~ ., data)
```

```
## # weights:  45 (28 variable)
## initial  value 195.552987
## iter  10 value 25.707822
## iter  20 value 4.439185
## iter  30 value 0.068485
## iter  40 value 0.000861
## final  value 0.000002
## converged
```

```
nlr_fit = predict(nlr_mod)
#nlr_fit
#Compute the confusion
M = matrix(0,3,3)
for (i in 1:178) {
  M[nlr_fit[i], wine[i,2]] = M[nlr_fit[i], wine[i,2]] + 1
}
M
```

```
##      [,1] [,2] [,3]
## [1,]   59    0    0
## [2,]    0   71    0
## [3,]    0    0   48
```

```
TP = diag(M)
FP = rowSums(M) - TP
FN = colSums(M) - TP
PREi = TP/(TP + FP)
RECi = TP/(TP + FN)
#compute the macro precision recall
PRE_ma = (1/3)*sum(PREi)
REC_ma = (1/3)*sum(RECi)
#compute the macro F measure
F_measure = 2*(PRE_ma*REC_ma)/(PRE_ma+REC_ma)
round(c(PRE_ma, REC_ma, F_measure),3)
```

```
## [1] 1 1 1
```

Comment: Since all number are on diag of the matrix and F measure is 1, these indicates that the model predicts perfectly.

1.2 (b) Linear discriminant

1.2.1 Using LDA

```r
library(MASS)
lda_mod = lda(Class ~ ., data)
lda_fit = predict(lda_mod, data)$class
M_lda = matrix(0,3,3)
for (i in 1:178) {
  M_lda[lda_fit[i], wine[i,2]] = M_lda[lda_fit[i], wine[i,2]] + 1
}
M_lda
```

```
##      [,1] [,2] [,3]
## [1,]   59    0    0
## [2,]    0   71    0
## [3,]    0    0   48
```

```r
TP = diag(M_lda)
FP = rowSums(M_lda) - TP
FN = colSums(M_lda) - TP
PREi = TP/(TP + FP)
RECi = TP/(TP + FN)
#compute the macro precision recall
PRE_lda = (1/3)*sum(PREi)
REC_lda = (1/3)*sum(RECi)
#compute the macro F measure
F_lda = 2*(PRE_lda*REC_lda)/(PRE_lda+REC_lda)
round(c(PRE_lda, REC_lda, F_lda),3)
```

```
## [1] 1 1 1
```

1.2.2 Using QDA

```r
qda_mod = qda(Class ~ ., data)
qda_fit = predict(qda_mod, data)$class
M_qda = matrix(0,3,3)
for (i in 1:178) {
  M_qda[qda_fit[i], wine[i,2]] = M_qda[qda_fit[i], wine[i,2]] + 1
}
M_qda
```

```
##      [,1] [,2] [,3]
## [1,]   59    1    0
## [2,]    0   70    0
## [3,]    0    0   48
```

```r
TP = diag(M_qda)
FP = rowSums(M_qda) - TP
FN = colSums(M_qda) - TP
```

```r
PREi = TP/(TP + FP)
RECi = TP/(TP + FN)
#compute the macro precision recall
PRE_qda = (1/3)*sum(PREi)
REC_qda = (1/3)*sum(RECi)
#compute the macro F measure
F_qda = 2*(PRE_qda*REC_qda)/(PRE_qda+REC_qda)
round(c(PRE_qda, REC_qda, F_qda),3)
```

```
## [1] 0.994 0.995 0.995
```

Comment: The result of using lda is as perfect as using multinom function and it has a F-measure equal to 1. For qda, there is misclassification and the F_measure less than 1.

1.3 (c) Using SVM

```r
library(e1071)
x = wine[,3:15]
y = as.factor(wine[,2])

model = svm(x, y, kernel = "radial")
pred = fitted(model)
#Compute the confusion
M_svm = matrix(0,3,3)
for (i in 1:178) {
  M_svm[pred[i], wine[i,2]] = M_svm[pred[i], wine[i,2]] + 1
}
M_svm
```

```
##      [,1] [,2] [,3]
## [1,]   59    0    0
## [2,]    0   71    0
## [3,]    0    0   48
```

```r
TP = diag(M_svm)
FP = rowSums(M_svm) - TP
FN = colSums(M_svm) - TP
PREi = TP/(TP + FP)
RECi = TP/(TP + FN)
#compute the macro precision recall
PRE_svm = (1/3)*sum(PREi)
REC_svm = (1/3)*sum(RECi)
#compute the macro F measure
F_svm = 2*(PRE_svm*REC_svm)/(PRE_svm+REC_svm)
round(c(PRE_svm, REC_svm, F_svm),3)
```

```
## [1] 1 1 1
```

Comment: The SVM performs perfectly and it has a F-measure equal to 1.

1.4 (d) Summarize

From (a)-(c) we can find that multinom, lda and svm predict the class perfectly and qda has a little mistake but still predicts most observations correct. Since discriminant analysis is good the distribution is kind of normal.

2 Questionv2 Simulation

2.1 (a) Show marginally independent

$$\begin{aligned}Cov(y,X_5) &= Cov(X_1\beta_1 + X_2\beta_2 + X_3\beta_3 + X_4\beta_4 - 4\sqrt{\rho}X_5\beta_5 + \epsilon, X_5)\\ &= \beta_1 Cov(X_1,X_5) + \beta_2 Cov(X_2,X_5) + \beta_3 Cov(X_3,X_5) + \beta_4 Cov(X_4,X_5) - 4\sqrt{\rho}\beta_5 Var(X_5)\end{aligned}$$

Since $X_k \sim N(0,1)$ and the correlations of all X_k except X_5 are ρ, while X_5 has the correlation $\sqrt{\rho}$ with all other p $-$ 1 variables. Beisdes, we have $\beta_i = 1$ for $i = 1, 2, ..., 5$.

Thus we have:

$$\begin{aligned}Cov(y,X_5) &= \beta_1 Cov(X_1,X_5) + \beta_2 Cov(X_2,X_5) + \beta_3 Cov(X_3,X_5) + \beta_4 Cov(X_4,X_5) - 4\sqrt{\rho}\beta_5 Var(X_5)\\ &= 4\sqrt{\rho} - 4\sqrt{\rho}\\ &= 0\end{aligned}$$

Since $Cov(y, X_5) = 0$ and both of them are normally distributed (i.e. uncorrelated = independent), we can say that X_5 is marginally independent of y.

2.2 (b) SIS & ISIS

```
library(glmnet)
```

```
## Loading required package: Matrix
## Loaded glmnet 3.0-2
```

```
library(MASS)
library(SIS)
p = 1500
n = 200
rho = 0.7
beta = c(1,1,1,1,-4*sqrt(rho),rep(0,p-5))
mu = rep(0,p)
sigma = matrix(rho, p, p) + diag(1 - rho, p)
sigma[,5] = sqrt(sigma[,5])
sigma[5,] = sqrt(sigma[5,])
```

```
##create a function for doing SIS ISIS once
sis_fun <- function() {
  x = mvrnorm(n, mu, sigma, tol = 1e-6, empirical = FALSE, EISPACK = FALSE)
  e = rnorm(n, 0, 1)
  y = x%*%beta + e
  model = SIS(x, y, family = 'gaussian', iter = FALSE)
  model_ite = SIS(x, y, family = 'gaussian', tune = 'bic', varISIS = 'aggr')

  X_model = x[,model$ix]
  X_model_ite = x[,model_ite$ix]
```

```r
  beta_new = as.vector(solve(t(X_model)%*%X_model)%*%t(X_model)%*%y)
  beta_new_ite = as.vector(solve(t(X_model_ite)%*%X_model_ite)%*%t(X_model_ite)%*%y)

  beta_hat = rep(0, p)
  beta_hat_ite = rep(0, p)
  beta_hat[model$ix] = beta_new
  beta_hat_ite[model_ite$ix] = beta_new_ite
  return(list(beta_hat, beta_hat_ite))
}

##to repeate 1000 times
sis_1000 = replicate(1000, sis_fun())
beta_sis = sis_1000[[1]]
beta_isis = sis_1000[[2]]
##since already get location 1 and 2's value, new index start from 3 and 4
ind1 = seq(from = 3, to = length(sis_1000), by = 2)
ind2 = seq(from = 4, to = length(sis_1000), by = 2)

for(i in ind1){
  beta_sis = rbind(beta_sis, sis_1000[[i]])
}

for(i in ind2){
  beta_isis = rbind(beta_isis, sis_1000[[i]])
}
##this takes a while so I write it out for future use
write.csv(beta_sis, "beta_sis.csv")
write.csv(beta_isis, "beta_isis.csv")
```

```r
beta_sis0 = read.csv("beta_sis.csv")
beta_isis0 = read.csv("beta_isis.csv")
```

```r
##Have to remove the first colum
beta_sis = beta_sis0[,-1]
beta_isis = beta_isis0[,-1]
dim(beta_sis)
```

```
## [1] 1000 1500
```

```r
dim(beta_isis)
```

```
## [1] 1000 1500
```

```r
##S: a number of selected covariates
S_sis = length(which(beta_sis != 0))/1000
S_isis = length(which(beta_isis != 0))/1000
S_sis
```

```
## [1] 9.998
```

```r
S_isis
```

```
## [1] 5.65
```

```r
##FN for SIS & ISIS
FN_sis = 0
for (i in 1:1000) {
  FN_sis1 = length(which(beta_sis[i,] == 0 && beta !=0))
  FN_sis = FN_sis + FN_sis1
}
FN_sis = FN_sis/1000
FN_sis
```

```
## [1] 0.26
```

```
FN_isis = 0
for (i in 1:1000) {
  FN_isis1 = length(which(beta_isis[i,] == 0 && beta !=0))
  FN_isis = FN_isis + FN_isis1
}
FN_isis = FN_isis/1000
FN_isis
```

```
## [1] 0.099
```

```
betah_sis = rep(0, 1500)
betah_isis = rep(0, 1500)
for (i in 1:1500) {
  betah_sis[i] = mean(beta_sis[,i])
  betah_isis[i] = mean(beta_isis[,i])
}
##B1
del1_sis = sum(abs(betah_sis - beta))
del1_isis = sum(abs(betah_isis - beta))

####B2
del2_sis = sqrt(sum((betah_sis - beta)^2))
del2_isis = sqrt(sum((betah_isis - beta)^2))

SIS = c(del1_sis, del2_sis, S_sis, FN_sis)
ISIS = c(del1_isis, del2_isis, S_isis, FN_isis)
result = data.frame(rbind(SIS, ISIS))
colnames(result) = c("B1", "B2", "#S", "#FN")
#result
```

Table 1: Simulation result for (b)

	$\|\Delta\beta\|_1$	$\|\Delta\beta\|_2$	#S	#FN
SIS	7.276	3.461	9.998	0.260
Iterated SIS	1.454	0.639	5.650	0.099

2.3 (c) Lasso, adaptive lasso, elastic net

```
p = 40
n = 200
mu = rep(0, p)
rho = 0.7
beta = c(1,1,1,1,-4*sqrt(rho),rep(0,p-5))
sigma = matrix(0, p, p)
for (j in 1:p) {
  for (k in 1:p) {
    sigma[j,k] = 0.5^(abs(j-k))
  }
}
```

2.3.1 Lasso

```
library(glmnet)
library(MASS)
lasso_fun <- function() {
  x = mvrnorm(n, mu, sigma, tol = 1e-6, empirical = FALSE, EISPACK = FALSE)
  e = rnorm(n, 0, 1)
  y = x%*%beta + e
  lasso = glmnet(x, y, family = "gaussian", alpha = 1)
  lasso_beta = lasso$beta
  obj_lasso = -deviance(lasso)
  k = lasso$df
  n = lasso$nobs
  BIC_lasso = log(n)*k - obj_lasso
  lambda_lasso = which.min(BIC_lasso)
  lasso_beta = lasso_beta[,lambda_lasso]
  return(lasso_beta)
}

lasso_1000 = replicate(1000, lasso_fun())
write.csv(t(lasso_1000), "beta_lasso.csv")
```

2.3.2 Adaptive lasso

```
library(glmnet)
library(MASS)
alasso_fun <- function() {
  x = mvrnorm(n, mu, sigma, tol = 1e-6, empirical = FALSE, EISPACK = FALSE)
  e = rnorm(n, 0, 1)
  y = x%*%beta + e
  beta1 = solve(t(x)%*%x)%*%(t(x)%*%y)
  w = 1/abs(as.vector(beta1))
  alasso = glmnet(x, y, family = "gaussian", alpha = 1, penalty.factor = w)
  alasso_beta = alasso$beta
  obj_alasso = -deviance(alasso)
  k = alasso$df
  n = alasso$nobs
  BIC_alasso = log(n)*k - obj_alasso
  lambda_alasso = which.min(BIC_alasso)
  alasso_beta = alasso_beta[,lambda_alasso]
  return(alasso_beta)
}

alasso_1000 = replicate(1000, alasso_fun())
write.csv(t(alasso_1000), "beta_alasso.csv")
```

2.3.3 Elastic net

```
library(glmnet)
library(MASS)
elastic_n_fun <- function() {
  x = mvrnorm(n, mu, sigma, tol = 1e-6, empirical = FALSE, EISPACK = FALSE)
  e = rnorm(n, 0, 1)
  y = x%*%beta + e

  elastic_n = glmnet(x, y, family = "gaussian", alpha = 0.5)
  elastic_n_beta = elastic_n$beta
  obj_elastic_n = -deviance(elastic_n)
  k = elastic_n$df
  n = elastic_n$nobs
  BIC_elastic_n = log(n)*k - obj_elastic_n
  lambda_elastic_n = which.min(BIC_elastic_n)
  elastic_n_beta = elastic_n_beta[,lambda_elastic_n]
  return(elastic_n_beta)
}
elastic_n_1000 = replicate(1000, elastic_n_fun())
write.csv(t(elastic_n_1000), "beta_elastic_n.csv")
```

```
beta_lasso0 = read.csv("beta_lasso.csv")
beta_alasso0 = read.csv("beta_alasso.csv")
beta_elastic_n0 = read.csv("beta_elastic_n.csv")

##Have to remove the first colum
beta_lasso = beta_lasso0[,-1]
beta_alasso = beta_alasso0[,-1]
beta_elastic_n = beta_elastic_n0[,-1]
```

```
##S: a number of selected covariates
S_lasso = length(which(beta_lasso != 0))/1000
S_alasso = length(which(beta_alasso != 0))/1000
S_elastic_n = length(which(beta_elastic_n != 0))/1000

##FN
FN_lasso = 0
for (i in 1:1000) {
  FN_lasso1 = length(which(beta_lasso[i,] == 0 && beta !=0))
  FN_lasso = FN_lasso + FN_lasso1
}
FN_lasso = FN_lasso/1000
FN_lasso
```

```
## [1] 0
```

```
FN_alasso = 0
for (i in 1:1000) {
  FN_alasso1 = length(which(beta_alasso[i,] == 0 && beta !=0))
  FN_alasso = FN_alasso + FN_alasso1
}
FN_alasso = FN_alasso/1000
FN_alasso
```

```
## [1] 0
```

```r
FN_elastic_n = 0
for (i in 1:1000) {
  FN_elastic_n1 = length(which(beta_elastic_n[i,] == 0 && beta !=0))
  FN_elastic_n = FN_elastic_n + FN_elastic_n1
}
FN_elastic_n = FN_elastic_n/1000
FN_elastic_n
```

```
## [1] 0
```

```r
betah_lasso = rep(0, 40)
betah_alasso = rep(0, 40)
betah_elastic_n = rep(0, 40)

for (i in 1:40) {
  betah_lasso[i] = mean(beta_lasso[,i])
  betah_alasso[i] = mean(beta_alasso[,i])
  betah_elastic_n[i] = mean(beta_elastic_n[,i])
}

##B1
del1_lasso = sum(abs(betah_lasso - beta))
del1_alasso = sum(abs(betah_alasso - beta))
del1_elastic_n = sum(abs(betah_elastic_n - beta))

##B2

del2_lasso = sqrt(sum((betah_lasso - beta)^2))
del2_alasso = sqrt(sum((betah_alasso - beta)^2))
del2_elastic_n = sqrt(sum((betah_elastic_n - beta)^2))

LASSO = c(del1_lasso, del2_lasso, S_lasso, FN_lasso)
ALASSO = c(del1_alasso, del2_alasso, S_alasso, FN_alasso)
ELANET = c(del1_elastic_n, del2_elastic_n, S_elastic_n, FN_elastic_n)
result = data.frame(rbind(LASSO, ALASSO, ELANET))
colnames(result) = c("B1", "B2", "#S", "#FN")
#result
```

Table 2: Simulation result for (c)

	$\|\Delta\beta\|_1$	$\|\Delta\beta\|_2$	#S	#FN
lasso	0.623	0.328	7.212	0
adaptive lasso	0.1297	0.0597	5.274	0
Elastic net ($\alpha = 0.5$)	0.791	0.424	9.666	0

2.4 (d) Summarize

In part (b) the delta beta of SIS is higher than ISIS and number of S of ISIS is closer to the real number. So ISIS did a better performance than SIS. In part (c) all delta beta are small and FN are all 0. The adaptive lasso has the better number of sumner of S and lasso is at the second place.

From the results of two tables, the delta beta of lasso group is less than SIS group. I think lasso, adaptive lasso and elastic net did a better job than SIS and ISIS in this case. This may because SIS contains noninformative variables and some informative variables are not selected.

3 Question 3

3.1 (a) Express as Gaussian graphical model

$$\begin{aligned}
P_{\mu,\Sigma}(x) &= \frac{1}{2\pi^{\frac{p}{2}} det(\Sigma)^{\frac{1}{2}}} exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\} \\
&= exp\{log(\frac{1}{2\pi^{\frac{p}{2}} det(\Sigma)^{\frac{1}{2}}}) - \frac{1}{2}(x^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu - \mu^T\Sigma^{-1}x + \mu^T\Sigma^{-1}\mu)\} \\
&= exp\{-\frac{1}{2}log((2\pi)^p det(\Sigma)) - \frac{1}{2}(x^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu - \mu^T\Sigma^{-1}x + \mu^T\Sigma^{-1}\mu)\}
\end{aligned}$$

Let $\Theta = \Sigma^{-1}$ and $\gamma = \Sigma^{-1}\mu$

$$\begin{aligned}
P_{\mu,\Sigma}(x) &= exp\{-\frac{1}{2}log((2\pi)^p det(\Theta^{-1})) - \frac{1}{2}(x^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu - \mu^T\Sigma^{-1}x + \mu^T\Sigma^{-1}\mu)\} \\
&= exp\{-\frac{1}{2}log(det(\frac{(2\pi)^p}{\Theta})) - \frac{1}{2}(x^T\Sigma^{-1}x - x^T\gamma - \gamma^T x + \mu^T\gamma)\} \\
&= exp\{\frac{1}{2}log(det(\frac{\Theta}{(2\pi)^p})) - \frac{1}{2}x^T\Sigma^{-1}x + \gamma^T x\} \\
&= exp\{\gamma^T x - \frac{1}{2}x^T\Sigma^{-1}x - (-\frac{1}{2}log(det(\frac{\Theta}{(2\pi)^p})))\}
\end{aligned}$$

Since $A(\Theta) = (-\frac{1}{2}log(det(\frac{\Theta}{(2\pi)^p})))$

$$\begin{aligned}
P_{\mu,\Sigma}(x) &= exp\{\gamma^T x - \frac{1}{2}x^T\Sigma^{-1}x - (-\frac{1}{2}log(det(\frac{\Theta}{(2\pi)^p})))\} \\
&= exp\{\gamma^T x - \frac{1}{2}x^T\Sigma^{-1}x - A(\Theta)\} \\
&= exp\{\sum_{s\in V}\gamma_s x_s - \frac{1}{2}\sum_{(s,t)\in E}\theta_{st}x_s x_t - A(\Theta)\}
\end{aligned}$$

3.2 (b)Comparisons glasso and neibourhood inference

3.2.1 lattice with 10

```r
library(XMRF)
lattice_fun <- function() {
  n = 400
  q = 10
  SIM = XMRF.Sim(n, q, model = "GGM", graph = "lattice")
  X = SIM$X
  Theta = SIM$B
  return(list(X, Theta))
}
structure = replicate(10, lattice_fun())
X = structure[[1]]
Theta = structure[[2]]

ind1 = seq(from = 3, to = length(structure), by = 2)
ind2 = seq(from = 4, to = length(structure), by = 2)
```

```r
for(i in ind1){
  X = X + structure[[i]]
}
for(i in ind2){
  Theta = Theta + structure[[i]]
}

X = X/10
Theta = Theta/10

library(glasso)
rho1 = 0.001
rho2 = 0.01
rho3 = 0.1

s = var(t(X))
Theta1_glasso = glasso(s, rho = rho1)$wi
Theta2_glasso = glasso(s, rho = rho2)$wi
Theta3_glasso = glasso(s, rho = rho3)$wi

Theta1_glasso[which(abs(Theta1_glasso)<0.1)] = 0
Theta2_glasso[which(abs(Theta2_glasso)<0.1)] = 0
Theta3_glasso[which(abs(Theta3_glasso)<0.1)] = 0

Theta1_glasso = Theta1_glasso - diag(diag(Theta1_glasso),10 ,10)
Theta2_glasso = Theta2_glasso - diag(diag(Theta2_glasso),10 ,10)
Theta3_glasso = Theta3_glasso - diag(diag(Theta3_glasso),10 ,10)

Spe1 = length(which(Theta1_glasso[which(Theta==0)]==0))/length(which(Theta==0))
Spe2 = length(which(Theta2_glasso[which(Theta==0)]==0))/length(which(Theta==0))
Spe3 = length(which(Theta3_glasso[which(Theta==0)]==0))/length(which(Theta==0))

Sen1 = length(which(Theta1_glasso[which(Theta!=0)]!=0))/length(which(Theta!=0))
Sen2 = length(which(Theta2_glasso[which(Theta!=0)]!=0))/length(which(Theta!=0))
Sen3 = length(which(Theta3_glasso[which(Theta!=0)]!=0))/length(which(Theta!=0))


CI = XMRF(X, method ="GGM", stability = "star", N = 100)
Theta_CI = CI$network[[5]]
Spe_CI = length(which(Theta_CI[which(Theta==0)]==0))/length(which(Theta==0))
Sen_CI = length(which(Theta_CI[which(Theta!=0)]!=0))/length(which(Theta!=0))

Spe_10_l = c(Spe1, Spe2, Spe3, Spe_CI)
Sen_10_l = c(Sen1, Sen2, Sen3, Sen_CI)
```

3.2.2 hub with 10

```r
library(XMRF)
hub_fun <- function() {
  n = 400
  q = 10
  SIM = XMRF.Sim(n, q, model = "GGM", graph = "hub")
  X = SIM$X
  Theta = SIM$B
  return(list(X, Theta))
}
structure = replicate(10, hub_fun())
X = structure[[1]]
Theta = structure[[2]]

ind1 = seq(from = 3, to = length(structure), by = 2)
ind2 = seq(from = 4, to = length(structure), by = 2)

for(i in ind1){
  X = X + structure[[i]]
}
for(i in ind2){
  Theta = Theta + structure[[i]]
}

X = X/10
Theta = Theta/10

rho1 = 0.001
rho2 = 0.01
rho3 = 0.1

s = var(t(X))
Theta1_glasso = glasso(s, rho = rho1)$wi
Theta2_glasso = glasso(s, rho = rho2)$wi
Theta3_glasso = glasso(s, rho = rho3)$wi

Theta1_glasso[which(abs(Theta1_glasso)<0.1)] = 0
Theta2_glasso[which(abs(Theta2_glasso)<0.1)] = 0
Theta3_glasso[which(abs(Theta3_glasso)<0.1)] = 0

Theta1_glasso = Theta1_glasso - diag(diag(Theta1_glasso),10 ,10)
Theta2_glasso = Theta2_glasso - diag(diag(Theta2_glasso),10 ,10)
Theta3_glasso = Theta3_glasso - diag(diag(Theta3_glasso),10 ,10)

Spe1 = length(which(Theta1_glasso[which(Theta==0)]==0))/length(which(Theta==0))
Spe2 = length(which(Theta2_glasso[which(Theta==0)]==0))/length(which(Theta==0))
Spe3 = length(which(Theta3_glasso[which(Theta==0)]==0))/length(which(Theta==0))

Sen1 = length(which(Theta1_glasso[which(Theta!=0)]!=0))/length(which(Theta!=0))
Sen2 = length(which(Theta2_glasso[which(Theta!=0)]!=0))/length(which(Theta!=0))
Sen3 = length(which(Theta3_glasso[which(Theta!=0)]!=0))/length(which(Theta!=0))

CI = XMRF(X, method ="GGM", stability = "star", N = 100)
Theta_CI = CI$network[[5]]

Spe_CI = length(which(Theta_CI[which(Theta==0)]==0))/length(which(Theta==0))
Sen_CI = length(which(Theta_CI[which(Theta!=0)]!=0))/length(which(Theta!=0))
Spe_10_h = c(Spe1, Spe2, Spe3, Spe_CI)
Sen_10_h = c(Sen1, Sen2, Sen3, Sen_CI)
```

3.2.3 lattice with 50

```r
library(XMRF)
lattice50_fun <- function() {
  n = 400
  q = 50
  SIM = XMRF.Sim(n, q, model = "GGM", graph = "lattice")
  X = SIM$X
  Theta = SIM$B
  return(list(X, Theta))
}
structure = replicate(10, lattice50_fun())
X = structure[[1]]
Theta = structure[[2]]

ind1 = seq(from = 3, to = length(structure), by = 2)
ind2 = seq(from = 4, to = length(structure), by = 2)

for(i in ind1){
  X = X + structure[[i]]
}
for(i in ind2){
  Theta = Theta + structure[[i]]
}

X = X/10
Theta = Theta/10

rho1 = 0.001
rho2 = 0.01
rho3 = 0.1

s = var(t(X))
Theta1_glasso = glasso(s, rho = rho1)$wi
Theta2_glasso = glasso(s, rho = rho2)$wi
Theta3_glasso = glasso(s, rho = rho3)$wi

Theta1_glasso[which(abs(Theta1_glasso)<0.1)] = 0
Theta2_glasso[which(abs(Theta2_glasso)<0.1)] = 0
Theta3_glasso[which(abs(Theta3_glasso)<0.1)] = 0

Theta1_glasso = Theta1_glasso - diag(diag(Theta1_glasso),50 ,50)
Theta2_glasso = Theta2_glasso - diag(diag(Theta2_glasso),50 ,50)
Theta3_glasso = Theta3_glasso - diag(diag(Theta3_glasso),50 ,50)

Spe1 = length(which(Theta1_glasso[which(Theta==0)]==0))/length(which(Theta==0))
Spe2 = length(which(Theta2_glasso[which(Theta==0)]==0))/length(which(Theta==0))
Spe3 = length(which(Theta3_glasso[which(Theta==0)]==0))/length(which(Theta==0))

Sen1 = length(which(Theta1_glasso[which(Theta!=0)]!=0))/length(which(Theta!=0))
Sen2 = length(which(Theta2_glasso[which(Theta!=0)]!=0))/length(which(Theta!=0))
Sen3 = length(which(Theta3_glasso[which(Theta!=0)]!=0))/length(which(Theta!=0))
```

```
CI = XMRF(X, method ="GGM", stability = "star", N = 100)
Theta_CI = CI$network[[5]]

Spe_CI = length(which(Theta_CI[which(Theta==0)]==0))/length(which(Theta==0))
Sen_CI = length(which(Theta_CI[which(Theta!=0)]!=0))/length(which(Theta!=0))

Spe_50_l = c(Spe1, Spe2, Spe3, Spe_CI)
Sen_50_l = c(Sen1, Sen2, Sen3, Sen_CI)
```

3.2.4 hub with 50

```
library(XMRF)
hub50_fun <- function() {
  n = 400
  q = 50
  SIM = XMRF.Sim(n, q, model = "GGM", graph = "hub")
  X = SIM$X
  Theta = SIM$B
  return(list(X, Theta))
}
structure = replicate(10, hub50_fun())
X = structure[[1]]
Theta = structure[[2]]

ind1 = seq(from = 3, to = length(structure), by = 2)
ind2 = seq(from = 4, to = length(structure), by = 2)

for(i in ind1){
  X = X + structure[[i]]
}
for(i in ind2){
  Theta = Theta + structure[[i]]
}

X = X/10
Theta = Theta/10

rho1 = 0.001
rho2 = 0.01
rho3 = 0.1

s = var(t(X))
Theta1_glasso = glasso(s, rho = rho1)$wi
Theta2_glasso = glasso(s, rho = rho2)$wi
Theta3_glasso = glasso(s, rho = rho3)$wi

Theta1_glasso[which(abs(Theta1_glasso)<0.1)] = 0
Theta2_glasso[which(abs(Theta2_glasso)<0.1)] = 0
Theta3_glasso[which(abs(Theta3_glasso)<0.1)] = 0

Theta1_glasso = Theta1_glasso - diag(diag(Theta1_glasso),50 ,50)
Theta2_glasso = Theta2_glasso - diag(diag(Theta2_glasso),50 ,50)
```

```
Theta3_glasso = Theta3_glasso - diag(diag(Theta3_glasso),50 ,50)

Spe1 = length(which(Theta1_glasso[which(Theta==0)]==0))/length(which(Theta==0))
Spe2 = length(which(Theta2_glasso[which(Theta==0)]==0))/length(which(Theta==0))
Spe3 = length(which(Theta3_glasso[which(Theta==0)]==0))/length(which(Theta==0))

Sen1 = length(which(Theta1_glasso[which(Theta!=0)]!=0))/length(which(Theta!=0))
Sen2 = length(which(Theta2_glasso[which(Theta!=0)]!=0))/length(which(Theta!=0))
Sen3 = length(which(Theta3_glasso[which(Theta!=0)]!=0))/length(which(Theta!=0))

CI = XMRF(X, method ="GGM", stability = "star", N = 100)
Theta_CI = CI$network[[5]]

Spe_CI = length(which(Theta_CI[which(Theta==0)]==0))/length(which(Theta==0))

Sen_CI = length(which(Theta_CI[which(Theta!=0)]!=0))/length(which(Theta!=0))
Spe_50_h = c(Spe1, Spe2, Spe3,Spe_CI)
Sen_50_h = c(Sen1, Sen2, Sen3, Sen_CI)
```

As the value of rho increases, the spesificity also increases. The sensitivity is 1 when rho is kind of small. There seems no significant difference between a lattice model and a hub model. when number of p increases, the sensitivity and specificity seem not change too much.

Table 3: Numerical results for the estimators of Θ

p	Model	Method	ρ	Estimator of Θ	
				Spe	Sen
10	Lattice	1	0.001	0.459	1
			0.010	0.838	1
			0.100	1	0
		2	$\times$	1	0.923
	Hub	1	0.001	0.381	1
			0.010	0.881	1
			0.100	1	0
		2	$\times$	1	1
50	Lattice	1	0.001	0.348	1
			0.010	0.947	1
			0.100	1	0
		2	$\times$	1	0.941
	Hub	1	0.001	0.333	1
			0.010	0.932	1
			0.100	1	0
		2	$\times$	0.333	1

Method 1: glasso Method 2: neighbourhood inference.