Advanced Data Analytics 2
Bioinformatics Project

Jaideep Kulkarni

Student ID: 110295747

1. Skeleton

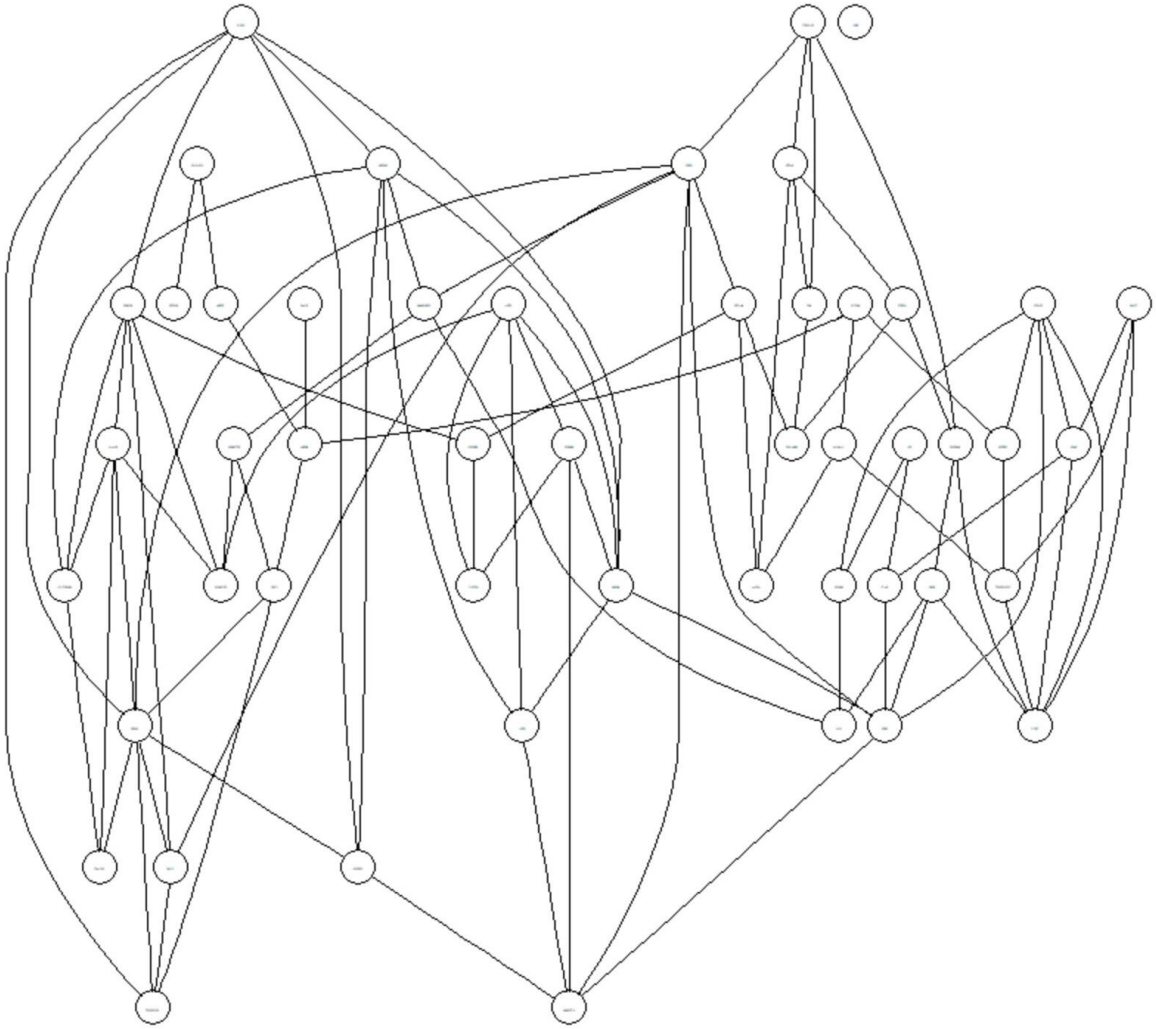

Fig. Skeleton of the gene regulatory network using gene expression data

Explanation

The Skeleton of the Directed Acyclic Structure is obtained. This is a kind of graph which contains edges but without any edge orientations. As the first step is completed, graph with no directions is the result. For every edge from node to node, constraints are checked if or not any conditioning set is present or if it is independent edge. Such set is called as Separation set. When such separation set is found, the edge between the nodes is deleted. There might be an occurrence of unshielded triplets i.e. the one node is connected to both the nodes. A node can be said as unshielded collider if two different nodes (which are not connected) are pointing towards one node. Final step would be to try to follow all the rules and avoiding any unshielded colliders or cycles which might be forbidden in Directed Acyclic Graph.

2. Top 10 Genes

```
> pcss
$G
     FIGF     LYVE1   CD300LG    SCARA5     PAMR1      SDPR     MYOM1     BTNL9    KCNIP2    SLC2A4
     TRUE      TRUE     FALSE      TRUE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE
    PDE2A       LEP    ACVR1C    ABCA10      AQP7    GPR146    ATP1A2     FXYD1  ARHGAP20      NPR1
    FALSE     FALSE     FALSE      TRUE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE
    ATOH8   ALDH1L1   ADAMTS5     RDH5      GPAM       CA4    KLHL29   GPIHBP1 LOC728264    MAMDC2
    FALSE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE
 TMEM132C     ITIH5     HSPB7     HSPB6       DMD     SPRY2    IGFBP6     CXCL2      EBF1       KLB
    FALSE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE      TRUE     FALSE
   CLEC3B   TMEM220      IBSP     HIF3A    IGSF10     CIDEC   C2orf40      LEPR   ANGPTL1
    FALSE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE      TRUE     FALSE

$zMin
 [1] 8.290130e+00 5.712266e+00 1.129263e+00 9.441702e+00 1.679557e+00 2.307439e-01 1.016270e+00
 [8] 1.476469e+00 6.494232e-01 2.661306e-01 1.904326e+00 1.810307e+00 1.343451e+00 5.118701e+00
[15] 5.317623e-01 6.062416e-01 1.143572e+00 3.438941e-01 6.163511e-01 1.578905e+00 1.579919e+00
[22] 1.395693e+00 6.705279e-01 1.440615e-01 1.740490e+00 1.921753e+00 4.255496e-04 1.418493e+00
[29] 1.905568e+00 1.210389e-01 4.390618e-01 3.597518e-01 8.198578e-01 1.202945e+00 6.797722e-01
[36] 1.309887e+00 1.278071e+00 9.299932e-01 1.233431e+01 1.928031e-01 7.253144e-01 1.657785e+00
[43] 6.131431e-01 1.534921e+00 5.818621e-01 6.828035e-01 1.339721e+00 2.632720e+00 4.214891e-01

> order(pcss$zMin)
 [1] 27 30 24 40  6 10 18 32 49 31 15 45 16 43 19  9 23 35 46 41 33 38  7  3 17 34 37 36 47 13 22 28
[33]  8 44 20 21 42  5 25 12 11 29 26 48 14  2  1  4 39
```

Fig. Genes and the order

By default, as the function “order” sort the values in ascending order, we consider the last 10 as the top 10 genes that have strong affect on the variable “ABCA9”. As we eliminated the “Class” variable there are only 49 variables to be considered. Top 10 variables which have strong effect on the single variable “ABCA9” are “HSBP7”, “SCARA5”, “FIGF”,“LYVE1”, ”LEPR”, ”ALDH1L1”, ”ABCA10”, ”CA4”, ”LEP”, ”PDE2A”.

3. Markov Blanket

```
> MB.Z <- learn.mb(mydata, "ABCA9", method="iamb", alpha=0.01)
> MB.Z
 [1] "EBF1"      "ABCA10"    "SCARA5"    "ACVR1C"    "CD300LG"   "LYVE1"     "GPAM"      "FIGF"
 [9] "LEPR"      "LOC728264" "TMEM132C"  "HIF3A"     "LEP"       "ANGPTL1"   "PAMR1"     "CLEC3B"
[17] "GPIHBP1"   "KLB"       "ATOH8"     "RDH5"      "NPR1"      "CIDEC"
```

Fig. Genes in the Markov Blanket

Incremental Association Markov Blanket (IAMB) is an algorithm which consists of two steps. In the first step, it adds predicted potential variables to the Markov Blanket and in second step, it removes the false positives which were added unintentionally in the first step.

Now we discretise the dataset. After discretising we follow to next question.

4. PC-simple Algorithm

```
[1] "FIGF"     "CD300LG"  "ARHGAP20" "KLHL29"   "SCARA5"   "MAMDC2"
[7] "CXCL2"    "ATP1A2"   "TMEM220"
```

Fig. Set of Parent and Children

Initially PC-simple takes in the dataset as an input and including predicted variables and the target variable. It produces PC, set of parents and children. Then PC-simple removes all the set of variables which does not contain any of the target parent or children variables by applying tests. Tests are done in the form of iteration i.e. step by step. Firstly, empty conditioning set and in first iteration of loop, the result would be the generation of the PC which does not contain the variables which are independent of the target parents and children. Firstly, let's say A0 is the first PC-simple. It makes A1 = A0 and starts the first iteration i.e. removal of the parents and children which are independent of the target node through A0. And then it updates to A1. Then A2 = A1 and tests are done on A1 and updated on A2 and so on. When an iteration has A(n-1) = An with no variables which are independent of the target node. That is the final PC set.

```
Naive Bayes

1212 samples
  50 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1091, 1091, 1091, 1091, 1090, 1091, ...
Resampling results across tuning parameters:

  usekernel  Accuracy  Kappa
  FALSE           NaN        NaN
   TRUE      0.975254  0.8278733

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held
 constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
```

Fig. Accuracy of the whole dataset

```
Naive Bayes

1212 samples
   9 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1091, 1091, 1091, 1091, 1090, 1091, ...
Resampling results across tuning parameters:

  usekernel  Accuracy  Kappa
  FALSE           NaN        NaN
   TRUE      0.986804  0.9166906

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held
 constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
```

Fig. Accuracy of just the set of Parent and Children set

As we can see, the Accuracy of the Parent and Children set is a little bit higher than the accuracy of the whole dataset.

Here we used 10-fold Cross Validation Method

Code

```
library(bnlearn)

library(pcalg)

library(readxl)

mydata <- read.csv("C:\\Users\\jaide\\Desktop\\data.csv")

View(mydata)


#dropping class variable

drops <- c("class")

mydata <- mydata[ , !(names(mydata) %in% drops)]


#code1

if (!requireNamespace("BiocManager", quietly = TRUE))
```

```
  install.packages("BiocManager")
BiocManager::install("Rgraphviz")
library(Rgraphviz)
mydata <- mydata[-51]
n <- nrow(mydata)
v <- colnames(mydata)
pc.fit <- skeleton(suffStat = list(C = cor(mydata), n=n), indepTest = gaussCItest, alpha = 0.01, labels = v)
plot(pc.fit, main = "Estimated graph")
#code1

#Markov Blanket
MB.Z <- learn.mb(mydata, "ABCA9", method="iamb", alpha=0.01)
MB.Z

for (r in 1:nrow(mydata))
  for (c in 1:ncol(mydata))
    if (mydata[r,c] > 304.7847){
      mydata[r,c] <- 1
    } else {
      mydata[r,c] <- 0
    }

print(mydata)
 View(mydata)

 library(bnlearn)
 global.network = si.hiton.pc(mydata, alpha=0.01)
 plot(global.network)

Cancer <- read.csv("C:\\Users\\jaide\\Desktop\\Cancer.csv")
```

```
library(bnlearn)
global.network = si.hiton.pc(Cancer, alpha=0.01)
plot(global.network)
HITON.PC.Class = learn.nbr(data, "Class", method="si.hiton.pc",alpha=0.01)
HITON.PC.Class

Cancer <- lapply(Cancer, as.numeric)
Cancer <- as.data.frame(Cancer)

library(pcalg)
library(binaryLogic)
library(e1071)

data <- as.data.frame(data)

data<-read.csv("C:\\Users\\jaide\\Desktop\\Cancer.csv", header=TRUE, sep=",")
data<-lapply(data, as.numeric)
data$Class=as.factor(data$Class)
nb_default <- naiveBayes(Class~., data)
nb_default
modelPred <- predict(nb_default, data)

modelPred
cMatrix <- table(modelPred, data$Class)
cMatrix

nb_pcset<-
naiveBayes(Class~FIGF+CD300LG+SCARA5+ATP1A2+ARHGAP20+KLHL29+MAMDC2+CXCL2+TMEM22
0,data)
check<-data[,-51]
check
fit <- train(data[,-51], data$Class, method = "nb",
```

```
        trControl = trainControl(method = "cv", number = 10))
fit
nb_pcset
model_pc <- predict(nb_pcset, data)
cMatrix <- table(model_pc, data$Class)
cMatrix


library(caret)


train_control <- trainControl(method="cv", number=10)
data
model <- train(Class~., data=data, trControl=train_control, method="nb",useKernel="True")
print(model)


library(caret)


folds <- 10
cvIndex <- createFolds(factor(data$Class), k=10, returnTrain = T)
train_control<-trainControl(index = cvIndex,method="cv",number=10)


nb.m1 <- train(
  x =
data[,c("FIGF","CD300LG","SCARA5","ATP1A2","ARHGAP20","KLHL29","MAMDC2","CXCL2","TMEM2
20")],
  y = data$Class,
  method = "nb",
  trControl = train_control
  )
nb.m1


confusionMatrix(nb.m1)
```

```
search_grid <- expand.grid(
  usekernel = FALSE,
  fL = 0:5,
  adjust = seq(0, 5, by = 1)
)

nb.m2 <- train(
  x = data[,-51],
  y = data$Class,
  method = "nb",
  trControl = train_control

)
nb.m2
```