

Project 1. Solution

Santiago Castro Dau, June Monge, Rachita Kumar, Sarah Lötscher

2/25/2022

Problem 1: Conditional independence and BNs

a)

The condition $A \perp B|C$ holds for this Bayesian network a).

Proof:

$$P(A, B, C) = P(A|C)P(B|C)P(C)$$

Using Bayes' rule, and substituting

$$\begin{aligned} P(A, B|C) &= \frac{P(A, B, C)}{P(C)} \\ &= \frac{P(A|C)P(B|C)P(C)}{P(C)} \\ &= P(A|C)P(B|C) \\ &\implies A \perp B|C \end{aligned}$$

Hence Proved.

In general, $A \perp B$ does not hold for this Bayesian network.

Proof:

Consider,

$$\begin{aligned} P(A, B) &= \sum_C P(A, B, C) \\ &= \sum_C P(A|C)P(B|C)P(C) \\ &= P(A)P(B) \sum_C \frac{P(C|B)P(C|A)}{P(C)} \\ &\implies A \perp B \text{ does not generally hold.} \end{aligned}$$

b)

The condition $A \perp B$ holds for this Bayesian network b).

Proof:

$$\begin{aligned}
 P(A, B, C) &= P(A)P(B)P(C|A, B) \\
 P(A, B) &= \sum_C P(A, B, C) \\
 &= \sum_C P(A)P(B)P(C|A, B) \\
 &= P(A)P(B) \sum_C P(C|A, B) \\
 &= P(A)P(B) \\
 &\implies A \perp B
 \end{aligned}$$

Hence Proved.

In general, $A \perp B|C$ does not hold for this Bayesian network.

Proof:

$$\begin{aligned}
 &\text{Consider,} \\
 P(A, B|C) &= \frac{P(A, B, C)}{P(C)} \\
 &= \frac{P(C|A, B)P(A)P(B)}{P(C)}
 \end{aligned}$$

Multiplying and dividing by $P(C|A)$ and using Bayes' rule,

$$\begin{aligned}
 &= \frac{P(C|A, B)P(A)P(B)}{P(C)} \frac{P(C|A)}{P(C|A)} \\
 &= \frac{P(C|A, B)P(A|C)P(B)}{P(C|A)}
 \end{aligned}$$

Multiplying and dividing by $P(C|B)P(C)$ and using Bayes' rule,

$$\begin{aligned}
 &= \frac{P(C|A, B)P(A|C)P(B)}{P(C|A)} \frac{P(C|B)}{P(C|B)} \\
 &= P(A|C)P(B|C) \frac{P(C|A, B)P(C)}{P(C|A)P(C|B)}
 \end{aligned}$$

Since $\frac{P(C|A, B)P(C)}{P(C|A)P(C|B)}$ will be equal to 1 in only rare cases, in general, $A \perp B|C$ does not hold for this Bayesian network. ## Problem 2: Markov blanket $MB(D) = \{B, F, E, G, C\}$ Consider,

$$P(D|A, B, C, E, F, G) = \frac{P(A, B, C, D, E, F, G)}{P(A, B, C, E, F, G)}$$

Since $A \perp B, C, D, E, F, G$

$$\begin{aligned}
 P(D|A, B, C, E, F, G) &= \frac{P(A)P(B, C, D, E, F, G)}{P(A)P(B, C, E, F, G)} \\
 &= P(D|MB(D))
 \end{aligned}$$

Problem 3: Learning Bayesian networks from protein data

a)

```

## Set seed
set.seed(2022)

## Read and index data
data <- vroom(file = "sachs.data.txt", show_col_types = FALSE)
data = data %>% rowid_to_column(var = 'row_id')

## Visualize data dimensions.
dim(data)

## [1] 853 12
• N = 853
• n = 11

## Split data
train = data %>% sample_frac(0.8, replace = FALSE)
test = anti_join(data, train, by = "row_id")

## Remove index column used for splitting
train = train %>% select(-row_id)
test = test %>% select(-row_id)

## Initializing score objects
score_object_train = scoreparameters(scoretype = 'bge', train)
score_object_test = scoreparameters(scoretype = 'bge', test)

```

b)

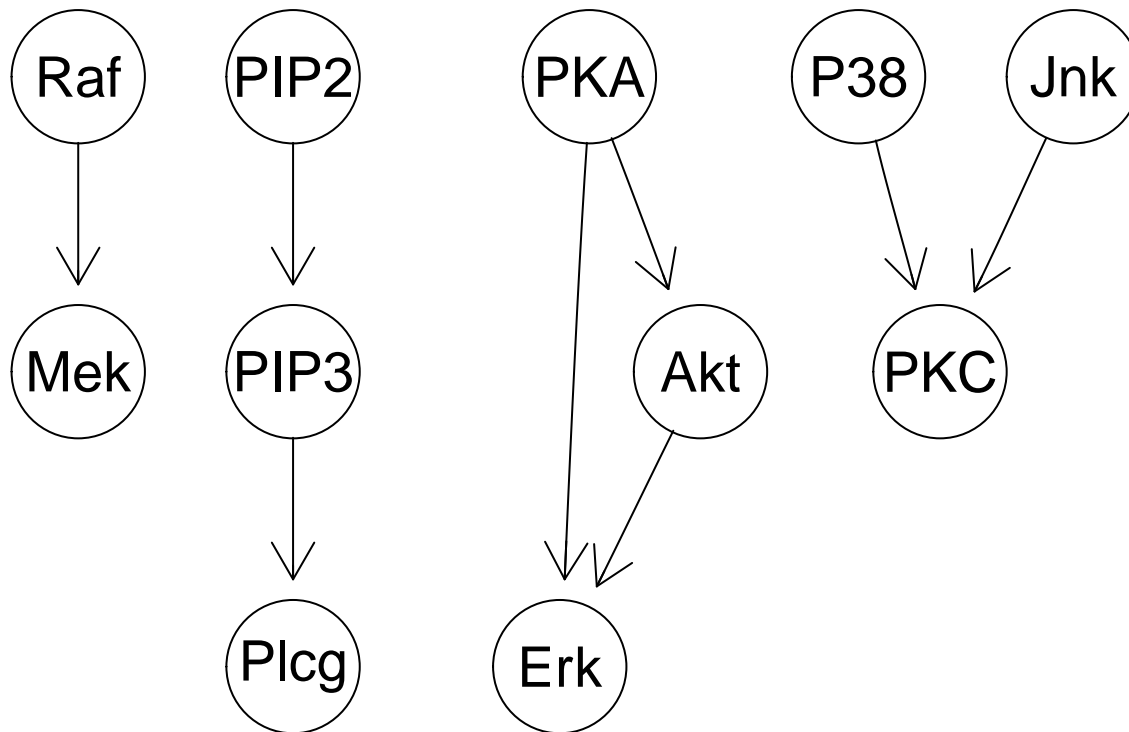
```

## Infer network on train data
inferred_network = orderMCMC(score_object_train)

## Construct graph object with the inferred network
graph_object = graphAM(adjMat = inferred_network$DAG,
                        edgemode='directed',
                        values=NA)

## Plot graph
plot(graph_object)

```



```
## Evaluate the log BGe score of the test data against the estimated DAG.
DAGscore(score_object_test, inferred_network$DAG)
```

```
## [1] -9456.304
```

c)

```
## Define function for average BGe score and average number of edges
avg_BGe_score_and_n_edges <- function(am) {

  ## Have to load libraries inside the function otherwise it can not find other
  ## functions even if they are already loaded.
  library(BiDAG)
  library(dplyr)

  ## Init objects to keep track of BGe score and number of edges
  n_edges = c()
  BGe_scores = c()

  for (i in 1:10) {
    ## Split data
    train = data %>% sample_frac(0.8, replace = FALSE)
    test = anti_join(data, train, by = "row_id")

    ## Remove id column used for splitting
    train = train %>% select(-row_id)
    test = test %>% select(-row_id)

    ## Initializing score objects
    score_object_train = scoreparameters(scoretype = 'bge',
                                          train,
```

```

                                bgepar = list(am = am, aw = NULL))
score_object_test = scoreparameters(scoretype = 'bge',
                                test,
                                bgepar = list(am = am, aw = NULL))

## Infer network on train data
inferred_network = orderMCMC(score_object_train)

## Append number of edges of the inferred network
n_edges = c(n_edges, sum(inferred_network$DAG))

## Append BGe score
BGe_scores = c(BGe_scores, DAGscore(score_object_test,
                                inferred_network$DAG))
}

## Return mean number of edges and mean BGe score
return(list("am_value" = am,
            "mean_n_edges" = mean(n_edges),
            "mean_BGe_score" = mean(BGe_scores)))
}

## Create list of alpha mu values (am values)
am_values = list(10e-5, 10e-3, 10e-1, 10, 10e2)

## Run the each average pair (for each am value) on parallel
## You might have to change this if you computer doesn't have 5 cores!!
results = mclapply(X = am_values,
                   FUN = avg_BGe_score_and_n_edges,
                   mc.cores = 5,
                   mc.preschedule = FALSE)

## Run this in case your OS is Windows ()
# results = lapply(X = am_values,
#                   FUN = avg_BGe_score_and_n_edges)

## Print results
do.call(rbind, results)

##      am_value mean_n_edges mean_BGe_score
## [1,] 1e-04      5.3         -9506.544
## [2,] 0.01      6.4         -9592.082
## [3,] 1         6.8         -9313.427
## [4,] 10        7          -9629.934
## [5,] 1000     17.4        -35617.53

## The closer x is to 0 the more negative the log(x) is. Therefore the least neg
## average log(BGe_score) corresponds to the largest BGe score (hence am = 1).

## Plot relearned DAG with whole data set
## Initializing score objects
score_object_data = scoreparameters(scoretype = 'bge',
                                data %>% select(-row_id),
                                bgepar = list(am = 1, aw = NULL))

```

```

## Infer network on train data
inferred_network = orderMCMC(score_object_data)

## Construct graph object with the inferred network
graph_object = graphAM(adjMat = inferred_network$DAG,
                        edgemode='directed',
                        values=NA)

## Plot graph
plot(graph_object)

```

