Advanced Machine Learning (732A96) Lab1

Anubhav Dikshit(anudi287)

16 September, 2019

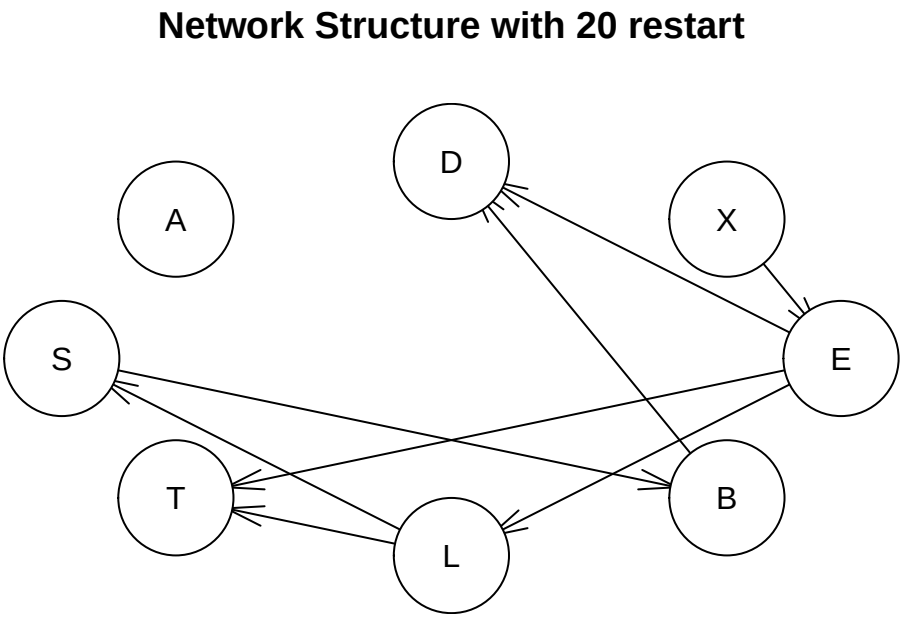
Contents

Questions

Questions 1

1) Show that multiple runs of the hill-climbing algorithm can return non-equivalent Bayesian network (BN) structures. Explain why this happens. Use the Asia dataset which is included in the bnlearn package. To load the data, run data("asia").

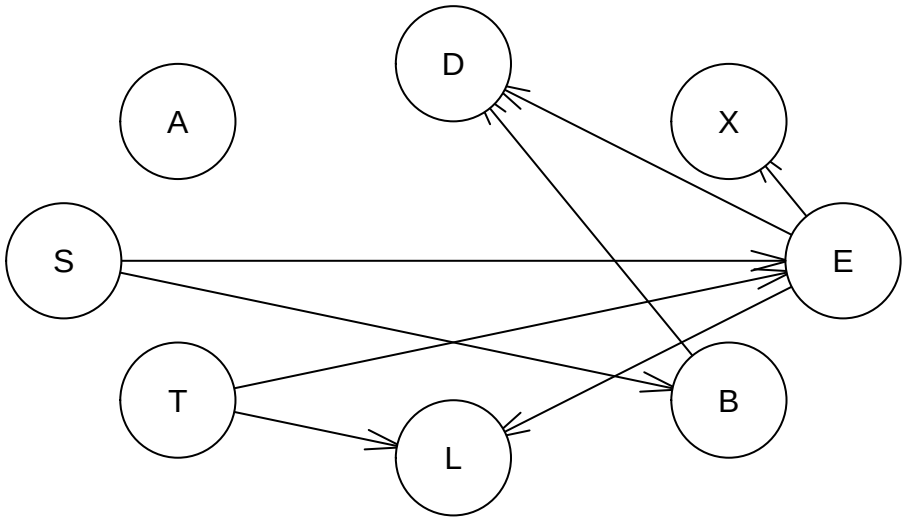
```
set.seed(12345)
asia_data <- bnlearn::asia

bayes_net = random.graph(colnames(asia))

hill_climbing_asia_1 <- hc(x=asia_data, start = bayes_net, restart = 20)
plot(hill_climbing_asia_1, main="Network Structure with 20 restart")
```




```
score(hill_climbing_asia_1, data=asia_data)
```

```
## [1] -11111.35
```

```
hill_climbing_asia_2 <- hc(x=asia_data, start = bayes_net, restart = 40)
plot(hill_climbing_asia_2, main="Network Structure with 40 restart")
```




```
score(hill_climbing_asia_2, data=asia_data)
```

```
## [1] -11115.83
```

```
all.equal(hill_climbing_asia_1,hill_climbing_asia_2)
```

```
## [1] "Different arc sets"
```

Questions 2

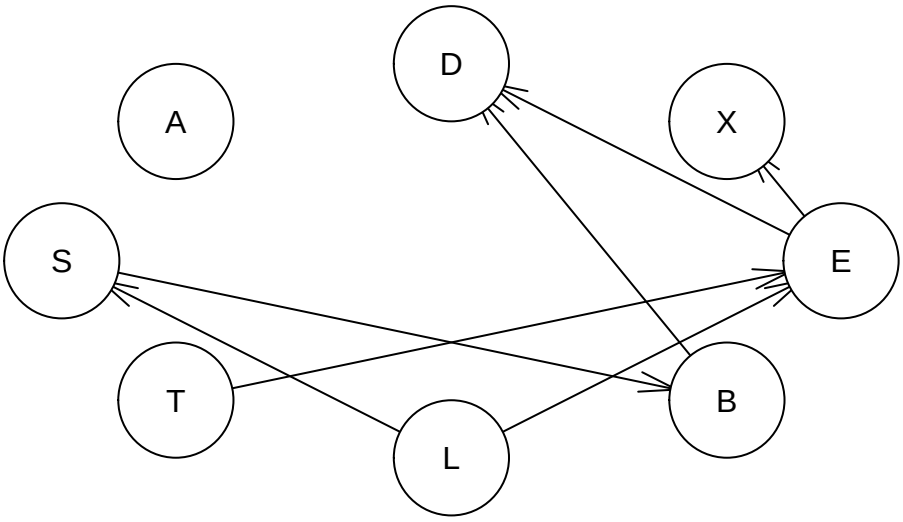
2) Learn a BN from 80 percent of the Asia dataset. The dataset is included in the bnlearn package. To load the data, run data("asia"). Learn both the structure and the parameters. Use any learning algorithm and settings that you consider appropriate. Use the BN learned to classify the remaining 20 percent of the Asia dataset in two classes: S = yes and S = no. In other words, compute the posterior probability distribution of S for each case and classify it in the most likely class. To do so, you have to use exact or approximate inference with the help of the bnlearn and gRain packages, i.e. you are not allowed to use functions such as predict. Report the confusion matrix, i.e. true/false positives/negatives. Compare your results with those of the true Asia BN, which can be obtained by running dag = model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]").

```
## set the seed to make your partition reproducible
set.seed(12345)
```

```r
asia_data <- bnlearn::asia

predict_from_network <- function(junc_tree, data, obs_variables, pred_variable) {
  prediction_fit <- rep(0,NROW(data))
  for (i in 1:NROW(data)) {
    X <- NULL
    for (j in obs_variables) {
      X[j] <- if(data[i, j] == "yes") "yes" else "no"
    }

    # Set evidence in junction tree for observation i
    # We have observations of all variables except:
    # S: If a person smokes or not
    evidence <- setEvidence(object = junc_tree,
                            nodes = obs_variables,
                            states = X)

    # Do prediction of S from junction tree with the above evidence
    prob_dist_fit <- querygrain(object = evidence, nodes = pred_variable)$S

    prediction_fit[i] <- if(prob_dist_fit["yes"] >= 0.5) "yes" else "no"
  }
  return(prediction_fit)
}


smp_size <- floor(0.80 * nrow(asia_data))

train_ind <- sample(seq_len(nrow(asia_data)), size = smp_size)
train <- asia_data[train_ind,]
test <- asia_data[-train_ind,]

# Learn BN-network structure
BN.structure <- hc(x = train, restart = 20)
score(BN.structure, train)
```

```
## [1] -8882.211
```

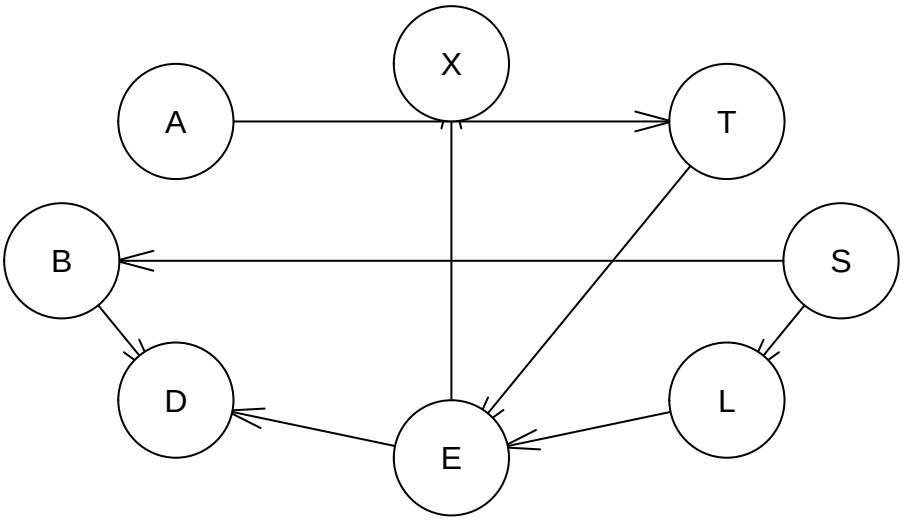
```r
plot(BN.structure, main="network structure of train")
```




```
BN.structure_true <- model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")
score(BN.structure_true, train)
```

```
## [1] -8883.927
```

```
plot(BN.structure_true, main="network structure based on rule")
```




```
# Fit parameters to train data
BN.fit <- bn.fit(x = BN.structure, data = train)
BN.fit_true <- bn.fit(x = BN.structure_true, data = train)

# Convert fit to gRain-object
BN.fit_gRain <- as.grain(BN.fit)
BN.fit_true_gRain <- as.grain(BN.fit_true)

# Compile BN
# Creating a junction tree (Lauritzen-Spiegelhalter algorithm) and establishing clique potentials
junc_tree <- compile(BN.fit_gRain)
junc_tree_true <- compile(BN.fit_true_gRain)

# Predict S from Bayesian Network and test data observations
prediction_fit <- predict_from_network(junc_tree = junc_tree, data = test,
                                       obs_variables = c("A", "T", "L", "B", "E", "X", "D"),
                                       pred_variable = c("S"))

prediction_fit_true <- predict_from_network(junc_tree = junc_tree_true, data = test,
                                            obs_variables = c("A", "T", "L", "B", "E", "X", "D"),
                                            pred_variable = c("S"))

# Calculate confusion matricies
confusion_matrix_fit <- table(prediction_fit, test$S)
confusion_matrix_fit
```

```
##
## prediction_fit  no yes
##            no  337 121
##            yes 176 366
```

```
confusion_matrix_true <- table(prediction_fit_true, test$S)
confusion_matrix_true
```

```
##
## prediction_fit_true  no yes
##                 no  337 121
##                 yes 176 366
```

Questions 3

3) In the previous exercise, you classified the variable S given observations for all the rest of the variables. Now, you are asked to classify S given observations only for the so-called Markov blanket of S, i.e. its parents plus its children plus the parents of its children minus S itself. Report again the confusion matrix.

```
MB_fit <- mb(x = BN.fit, node = c("S"))
MB_fit_true <- mb(x = BN.fit_true, node = c("S"))

prediction_fit_MB <- predict_from_network(junc_tree = junc_tree, data = test,
                                          obs_variables = MB_fit,
                                          pred_variable = c("S"))

prediction_fit_true_MB <- predict_from_network(junc_tree = junc_tree_true, data = test,
                                               obs_variables = MB_fit_true,
                                               pred_variable = c("S"))

# Calculate confusion matricies
confusion_matrix_fit <- table(prediction_fit, test$S)
confusion_matrix_fit
```

```
##
## prediction_fit  no yes
##            no  337 121
##            yes 176 366
```

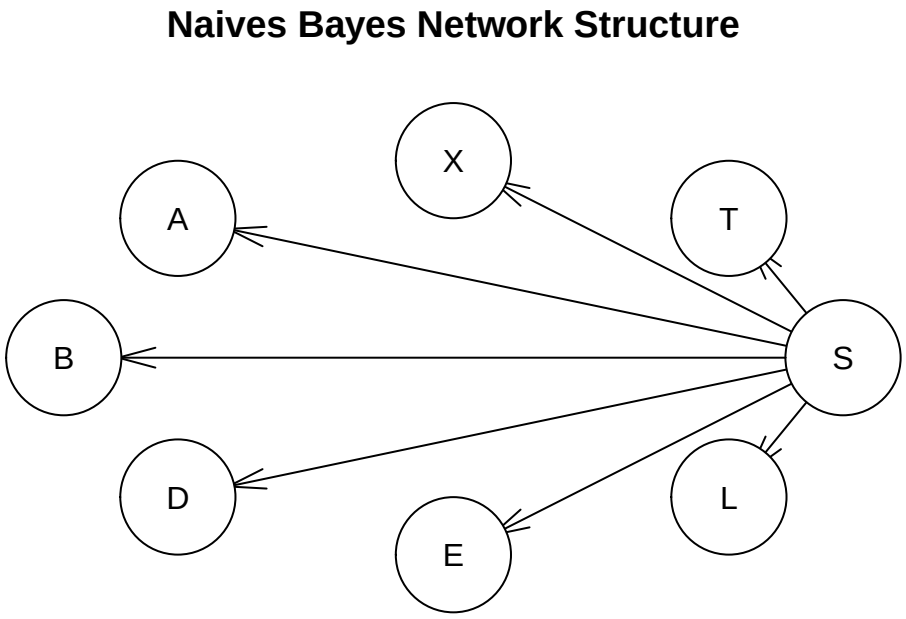
```
confusion_matrix_fit_true <- table(prediction_fit_true, test$S)
confusion_matrix_fit_true
```

```
##
## prediction_fit_true  no yes
##                 no  337 121
##                 yes 176 366
```

Questions 4

4) Repeat the exercise (2) using a naive Bayes classifier, i.e. the predictive variables are independent given the class variable. See p. 380 in Bishop's book or Wikipedia for more information on the naive Bayes classifier. Model the naive Bayes classifier as a BN. You have to create the BN by hand, i.e. you are not allowed to use the function naive.bayes from the bnlearn package.

```
naive_bayes_structure <- model2network("[S][A|S][T|S][L|S][B|S][E|S][X|S][D|S]")

# Fit parameters of network to train data
BN.fit_naive_bayes <- bn.fit(x = naive_bayes_structure, data = test)
plot(naive_bayes_structure, main="Naives Bayes Network Structure")
```




```
score(naive_bayes_structure, test)
```

```
## [1] -2954.038
```

```
# Convert fit to gRain-object
BN.fit_naive_bayes_grain <- as.grain(BN.fit_naive_bayes)

# Generate juncion tree and clique potentials
junc_tree_naive_bayes <- compile(BN.fit_naive_bayes_grain)

prediction_fit_naive_bayes <- predict_from_network(junc_tree = junc_tree_naive_bayes, data = test,
```

```
                                                    obs_variables = c("A", "T", "L", "B", "E", "X", "D"),
                                                    pred_variable = c("S"))

prediction_fit_true <- predict_from_network(junc_tree = junc_tree_true, data = test,
                                            obs_variables = c("A", "T", "L", "B", "E", "X", "D"),
                                            pred_variable = c("S"))

# Calculate confusion matricies
confusion_matrix_naive_bayes <- table(prediction_fit_naive_bayes, test$S)
confusion_matrix_naive_bayes
```

```
##
## prediction_fit_naive_bayes  no yes
##                         no  359 180
##                         yes 154 307
```

```
confusion_matrix_fit_true <- table(prediction_fit_true, test$S)
confusion_matrix_fit_true
```

```
##
## prediction_fit_true  no yes
##                  no  337 121
##                  yes 176 366
```

Questions 5

5) Explain why you obtain the same or different results in the exercises (2-4).

Answer: We can see that results of 2 and 3 are the same from the confusion table, this is expected since essentially its the same model. 2 and 4 are different clearly because of the network structure.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
options(scipen=999)

library("tidyverse") #ggplot and dplyr
library("gridExtra") # combine plots
library("knitr") # for pdf
library("bnlearn") # ADM
library("gRain") # ADM


if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
#BiocManager::install("gRain")


# The palette with black:
```

```r
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73",
                "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
set.seed(12345)
set.seed(12345)
asia_data <- bnlearn::asia

bayes_net = random.graph(colnames(asia))

hill_climbing_asia_1 <- hc(x=asia_data, start = bayes_net, restart = 20)
plot(hill_climbing_asia_1, main="Network Structure with 20 restart")
score(hill_climbing_asia_1, data=asia_data)

hill_climbing_asia_2 <- hc(x=asia_data, start = bayes_net, restart = 40)
plot(hill_climbing_asia_2, main="Network Structure with 40 restart")
score(hill_climbing_asia_2, data=asia_data)

all.equal(hill_climbing_asia_1,hill_climbing_asia_2)


## set the seed to make your partition reproducible
set.seed(12345)

asia_data <- bnlearn::asia

predict_from_network <- function(junc_tree, data, obs_variables, pred_variable) {
  prediction_fit <- rep(0,NROW(data))
  for (i in 1:NROW(data)) {
    X <- NULL
    for (j in obs_variables) {
      X[j] <- if(data[i, j] == "yes") "yes" else "no"
    }

    # Set evidence in junction tree for observation i
    # We have observations of all variables except:
    # S: If a person smokes or not
    evidence <- setEvidence(object = junc_tree,
                            nodes = obs_variables,
                            states = X)

    # Do prediction of S from junction tree with the above evidence
    prob_dist_fit <- querygrain(object = evidence, nodes = pred_variable)$S

    prediction_fit[i] <- if(prob_dist_fit["yes"] >= 0.5) "yes" else "no"
  }
  return(prediction_fit)
}


smp_size <- floor(0.80 * nrow(asia_data))

train_ind <- sample(seq_len(nrow(asia_data)), size = smp_size)
train <- asia_data[train_ind,]
test <- asia_data[-train_ind,]
```

```r
# Learn BN-network structure
BN.structure <- hc(x = train, restart = 20)
score(BN.structure, train)
plot(BN.structure, main="network structure of train")

BN.structure_true <- model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")
score(BN.structure_true, train)
plot(BN.structure_true, main="network structure based on rule")

# Fit parameters to train data
BN.fit <- bn.fit(x = BN.structure, data = train)
BN.fit_true <- bn.fit(x = BN.structure_true, data = train)

# Convert fit to gRain-object
BN.fit_gRain <- as.grain(BN.fit)
BN.fit_true_gRain <- as.grain(BN.fit_true)

# Compile BN
# Creating a junction tree (Lauritzen-Spiegelhalter algorithm) and establishing clique potentials
junc_tree <- compile(BN.fit_gRain)
junc_tree_true <- compile(BN.fit_true_gRain)

# Predict S from Bayesian Network and test data observations
prediction_fit <- predict_from_network(junc_tree = junc_tree, data = test,
                                       obs_variables = c("A", "T", "L", "B", "E", "X", "D"),
                                       pred_variable = c("S"))

prediction_fit_true <- predict_from_network(junc_tree = junc_tree_true, data = test,
                                            obs_variables = c("A", "T", "L", "B", "E", "X", "D"),
                                            pred_variable = c("S"))

# Calculate confusion matricies
confusion_matrix_fit <- table(prediction_fit, test$S)
confusion_matrix_fit

confusion_matrix_true <- table(prediction_fit_true, test$S)
confusion_matrix_true



MB_fit <- mb(x = BN.fit, node = c("S"))
MB_fit_true <- mb(x = BN.fit_true, node = c("S"))

prediction_fit_MB <- predict_from_network(junc_tree = junc_tree, data = test,
                                          obs_variables = MB_fit,
                                          pred_variable = c("S"))

prediction_fit_true_MB <- predict_from_network(junc_tree = junc_tree_true, data = test,
                                               obs_variables = MB_fit_true,
                                               pred_variable = c("S"))

# Calculate confusion matricies
confusion_matrix_fit <- table(prediction_fit, test$S)
```

```r
confusion_matrix_fit

confusion_matrix_fit_true <- table(prediction_fit_true, test$S)
confusion_matrix_fit_true

naive_bayes_structure <- model2network("[S][A|S][T|S][L|S][B|S][E|S][X|S][D|S]")

# Fit parameters of network to train data
BN.fit_naive_bayes <- bn.fit(x = naive_bayes_structure, data = test)
plot(naive_bayes_structure, main="Naives Bayes Network Structure")
score(naive_bayes_structure, test)

# Convert fit to gRain-object
BN.fit_naive_bayes_grain <- as.grain(BN.fit_naive_bayes)

# Generate juncion tree and clique potentials
junc_tree_naive_bayes <- compile(BN.fit_naive_bayes_grain)

prediction_fit_naive_bayes <- predict_from_network(junc_tree = junc_tree_naive_bayes, data = test,
                                                   obs_variables = c("A", "T", "L", "B", "E", "X", "D")
                                                   pred_variable = c("S"))

prediction_fit_true <- predict_from_network(junc_tree = junc_tree_true, data = test,
                                            obs_variables = c("A", "T", "L", "B", "E", "X", "D"),
                                            pred_variable = c("S"))

# Calculate confusion matricies
confusion_matrix_naive_bayes <- table(prediction_fit_naive_bayes, test$S)
confusion_matrix_naive_bayes

confusion_matrix_fit_true <- table(prediction_fit_true, test$S)
confusion_matrix_fit_true
```