Advanced Machine Learning - Lab1

Thijs Quast (thiqu264)

19 September, 2019

Contents

Assignment 1

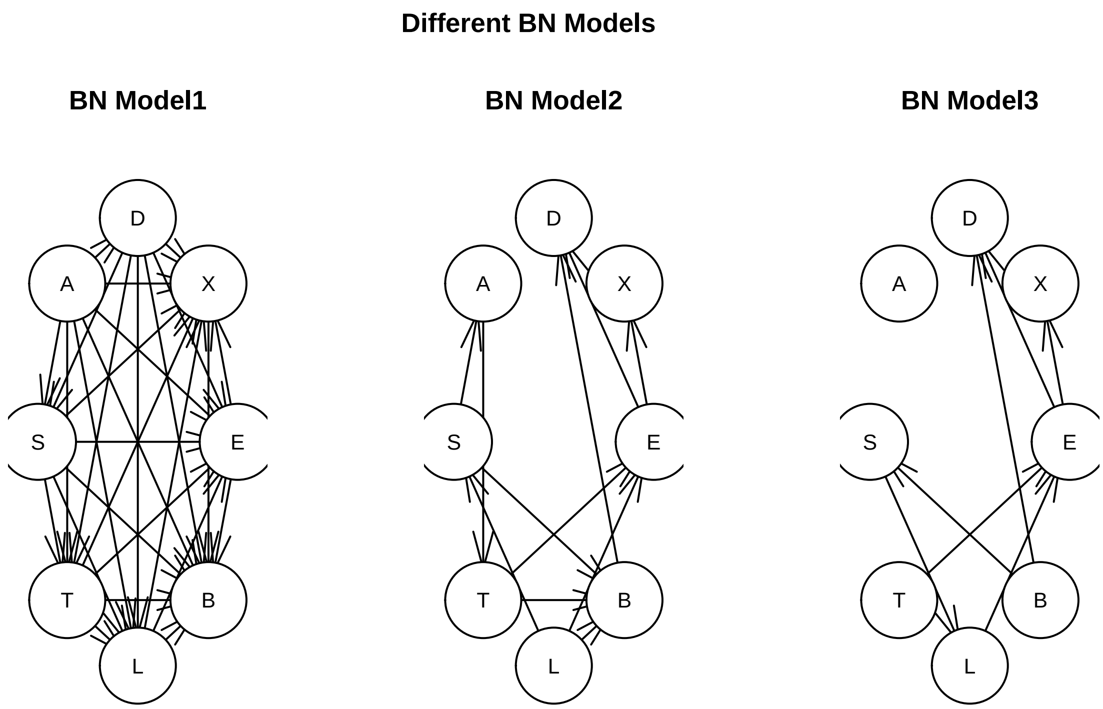
```r
library(bnlearn)
```

```
##
## Attaching package: 'bnlearn'
## The following object is masked from 'package:stats':
##
##     sigma
```

```r
data("asia")
```

```r
model1 <- hc(asia, start = NULL, restart = 100, score = "loglik")
model2 <- hc(asia, start = NULL, restart = 50, score = "aic")
model3 <- hc(asia, start = NULL, restart = 10, score = "bic")
```

```r
par(mfrow=c(1,3), oma=c(0,0,2,0))
plot(model1, main="BN Model1")
plot(model2, main="BN Model2")
plot(model3, main="BN Model3")
title(main="Different BN Models", outer = T)
```




```r
arcs(model1)
```

```
##      from to
## [1,] "E"  "X"
## [2,] "T"  "B"
```

```
##  [3,] "A"  "E"
##  [4,] "X"  "B"
##  [5,] "L"  "E"
##  [6,] "A"  "D"
##  [7,] "D"  "B"
##  [8,] "T"  "E"
##  [9,] "S"  "B"
## [10,] "A"  "L"
## [11,] "D"  "X"
## [12,] "D"  "E"
## [13,] "E"  "B"
## [14,] "A"  "X"
## [15,] "S"  "T"
## [16,] "L"  "X"
## [17,] "D"  "T"
## [18,] "L"  "B"
## [19,] "D"  "L"
## [20,] "T"  "X"
## [21,] "D"  "S"
## [22,] "T"  "L"
## [23,] "A"  "T"
## [24,] "A"  "B"
## [25,] "S"  "X"
## [26,] "S"  "L"
## [27,] "A"  "S"
## [28,] "S"  "E"
```

```r
arcs(model2)
```

```
##       from to
##  [1,] "L"  "B"
##  [2,] "S"  "A"
##  [3,] "S"  "B"
##  [4,] "E"  "D"
##  [5,] "B"  "D"
##  [6,] "L"  "S"
##  [7,] "T"  "B"
##  [8,] "T"  "E"
##  [9,] "L"  "E"
## [10,] "A"  "T"
## [11,] "E"  "X"
```

```r
arcs(model3)
```

```
##      from to
## [1,] "L"  "E"
## [2,] "B"  "S"
## [3,] "E"  "X"
## [4,] "T"  "E"
## [5,] "B"  "D"
## [6,] "E"  "D"
## [7,] "S"  "L"
```

```r
vstructs(model1)
```

```
##      X Z Y
```

```r
vstructs(model2)
```

```
##      X   Z   Y  
## [1,] "S" "B" "T"
## [2,] "S" "B" "L"
## [3,] "T" "B" "L"
## [4,] "T" "E" "L"
## [5,] "B" "D" "E"
```

```r
vstructs(model3)
```

```
##      X   Z   Y  
## [1,] "T" "E" "L"
## [2,] "B" "D" "E"
```

```r
all.equal(model1, model2, model3)
```

```
## [1] "Different number of directed/undirected arcs"
```

As can be seen above, multiple runs of the hill-climbing algorithm, with e.g. different score settings result in different Bayesian Networks. Firstly this can seen from the plotted graphs, also the arcs are different. In addition, when I use the all.equal function in R, it returns that the models are different and how they are different: "Different number of directed/undirected arcs".

Probably, different starting points of the algorithm with respect to the order of letters return different Bayesian Networks.

Assignment 2

```r
library(RBGL)
```

```
## Loading required package: graph

## Loading required package: BiocGenerics

## Loading required package: parallel

## 
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
## 
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:bnlearn':
## 
##     path, score

## The following objects are masked from 'package:stats':
## 
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
## 
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colMeans, colnames, colSums, dirname, do.call, duplicated,
```

```
##     eval, evalq, Filter, Find, get, grep, grepl, intersect,
##     is.unsorted, lapply, lengths, Map, mapply, match, mget, order,
##     paste, pmax, pmax.int, pmin, pmin.int, Position, rank, rbind,
##     Reduce, rowMeans, rownames, rowSums, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max,
##     which.min

##
## Attaching package: 'graph'

## The following objects are masked from 'package:bnlearn':
##
##     degree, nodes, nodes<-
```

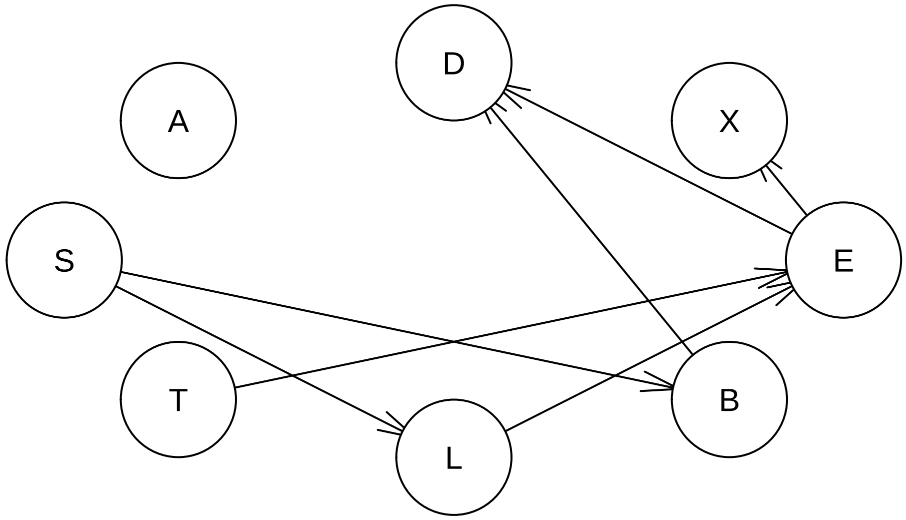
```
library(gRain)
```

```
## Loading required package: gRbase

##
## Attaching package: 'gRbase'

## The following objects are masked from 'package:bnlearn':
##
##     ancestors, children, parents
```

```
#train and test split, learned in Machine Learning course
n <- dim(asia)[1]
set.seed(12345)
id <- sample(1:n, floor(n*0.8))
train <- asia[id,]

id1 <- setdiff(1:n, id)
set.seed(12345)
id2 <- sample(id1, floor(n*0.2))
test <- asia[id2,]
```

```
# Use exact inference
# Create structure
structure <- hc(train)
fit <- bn.fit(x = structure, data = train)
fit_grain <- as.grain(fit)
```

```
plot(structure)
```




```
compiled_grain <- compile(fit_grain)
```

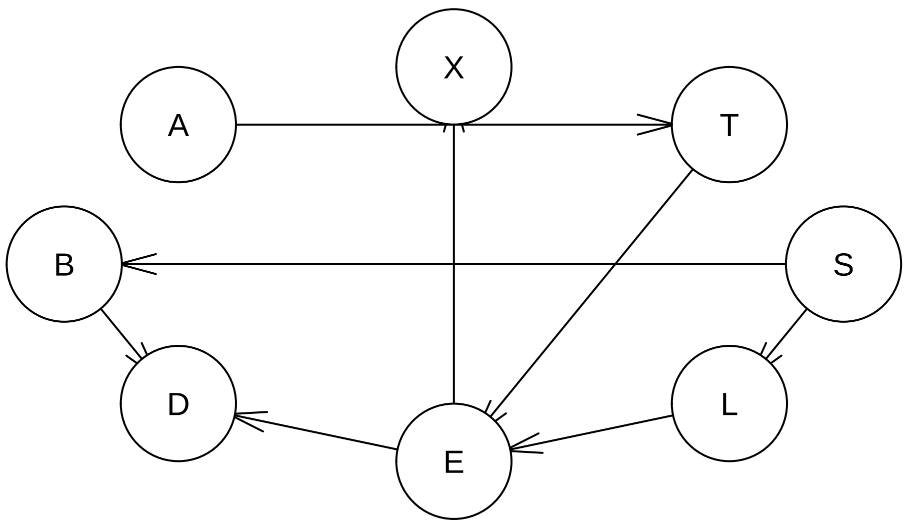
```
# Manipulating data
# The function querygrain needs the data to be in character form
test2 <- test
test <- apply(test, 2, as.character)
```

```
predictions <- c()

for (i in 1:nrow(test)){
  evidence <- setFinding(object = compiled_grain,
                                  nodes = c("A", "T", "L", "B", "E", "X", "D"),
                                  states = test[i, -2])

  posterior <- unlist(querygrain(object = evidence, nodes="S"))

  if (posterior[1] > 0.5) {
    predictions[i] <- "No"
  } else {
    predictions[i] <- "Yes"
  }

}
```

```
confusion_matrix <- table(test2$S, predictions)
confusion_matrix
```

```
##      predictions
##         No Yes
##   no  322 146
##   yes 120 412
```

```r
# True Bayesian Network
dag <- model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")
fit_true <- bn.fit(x = dag, data = train)
fit_true <- as.grain(fit_true)
plot(dag)
```




```r
compile_true <- compile(fit_true)
```

```r
predictions_true <- c()

for (i in 1:nrow(test)){
  evidence <- setFinding(object = compile_true,
                                  nodes = c("A", "T", "L", "B", "E", "X", "D"),
                                  states = test[i, -2])

  posterior <- unlist(querygrain(object = evidence, nodes="S"))

  if (posterior[1] > 0.5) {
    predictions_true[i] <- "No"
  } else {
    predictions_true[i] <- "Yes"
  }
```

```
}
```

```
confusion_true <- table(test2$S, predictions_true)
confusion_true
```

```
##      predictions_true
##        No Yes
##   no  322 146
##   yes 120 412
```

```
error_true <- (confusion_true[1,2] + confusion_true[2,1])/(sum(confusion_true))
error_true
```

```
## [1] 0.266
```

Assignment 3

```
markov_blanket <- mb(x = fit, node = "S")
markov_blanket
```

```
## [1] "L" "B"
```

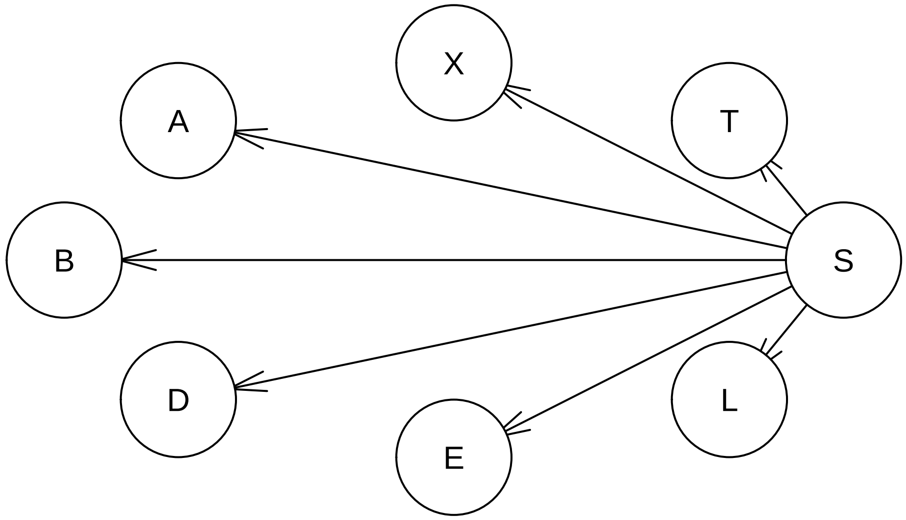
```
predictions_mb <- c()

for (i in 1:nrow(test)){
  evidence <- setFinding(object = compiled_grain,
                                 nodes = markov_blanket,
                                 states = test[i, markov_blanket])

  posterior <- unlist(querygrain(object = evidence, nodes="S"))

  if (posterior[1] > 0.5) {
    predictions_mb[i] <- "No"
  } else {
    predictions_mb[i] <- "Yes"
  }

}
```

```
confusion_matrix_mb <- table(test2$S, predictions_mb)
confusion_matrix_mb
```

```
##      predictions_mb
##        No Yes
##   no  322 146
##   yes 120 412
```

Assignment 4

```
# Naive Bayes:
naive_bayes = model2network("[S][A|S][T|S][L|S][B|S][E|S][X|S][D|S]")
plot(naive_bayes)
```




```
naive_bayes <- bn.fit(x = naive_bayes, data = train)
naive_bayes <- as.grain(naive_bayes)
naive_bayes <- compile(naive_bayes)
```

```
naive_predictions <- c()

for (i in 1:nrow(test)){
  evidence <- setFinding(object = naive_bayes,
                                  nodes = c("A", "T", "L", "B", "E", "X", "D"),
                                  states = test[i, -2])

  posterior <- unlist(querygrain(object = evidence, nodes="S"))

  if (posterior[1] > 0.5) {
    naive_predictions[i] <- "No"
  } else {
    naive_predictions[i] <- "Yes"
  }

}
```

```
confusion_naive_bayes <- table(test2$S, naive_predictions)
confusion_naive_bayes
```

```
##      naive_predictions
##         No Yes
##   no   349 119
```

```
##   yes 188 344
```

Assignment 5

```r
# Trained model (Assignment 2)
confusion_matrix
```

```
##      predictions
##        No Yes
##   no  322 146
##   yes 120 412
```

```r
error <- (confusion_matrix[1,2] + confusion_matrix[2,1])/(sum(confusion_matrix))
error
```

```
## [1] 0.266
```

```r
# True model (Assignment 2)
confusion_true
```

```
##      predictions_true
##        No Yes
##   no  322 146
##   yes 120 412
```

```r
error_true
```

```
## [1] 0.266
```

```r
# Markov blanket (Assignment 3)
confusion_matrix_mb
```

```
##      predictions_mb
##        No Yes
##   no  322 146
##   yes 120 412
```

```r
error_mb <- (confusion_matrix_mb[1,2] + confusion_matrix_mb[2,1])/(sum(confusion_matrix_mb))
error_mb
```

```
## [1] 0.266
```

```r
# Naive bayes (Assignment 4)
confusion_naive_bayes
```

```
##      naive_predictions
##        No Yes
##   no  349 119
##   yes 188 344
```

```r
error_naive_bayes <- (confusion_naive_bayes[1,2] +   confusion_naive_bayes[2,1])/(sum(confusion_naive_ba

error_naive_bayes
```

```
## [1] 0.307
```

The models from questions 2 and 3 return exactly the same results. Probably, conditioning on the Markov Blanket is already sufficient to construct the model. Therefore more elaborate models are not better in performance.

The fact that the Naive Bayes classifier performs worse than the other models is because in this model we assume independence amongst all explanatory variables. In practice this is very unlikely, therefore it is logical that the Naive Bayes classifier performs worse than the other models.