LAB1

Arash Haratian, Daniel Díaz-Roncero González, Elena Dalla Torre & Juan Manuel Pardo Ladino

2023-09-06

Question 1

Two DAGs are equivalent when they represent the same independencies. Meaning, when: 1) The two graphs have the same skeleton; 2) The two graphs have the same unshielded colliders.

The Hill-Climbing algorithm is a heuristic algorithm that uses the Bayesian score to construct a graph starting from an empty graph. Then edges are added/modified/removed based on how much they improve the Bayesian score.

The HC algorithm is not asymptotically correct. This means that even under the faithfulness assumption it may not return the true graph as it can get stuck in a local minimum. Consequently, it can return non-equivalent Bayesian networks.

```r
# load the data
data("asia")
```

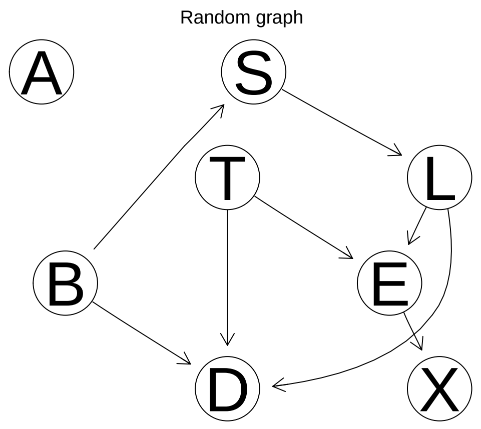
In order to get two different graphs from the hc() function in the bnlearn package we first change the initial graph. In the first run of hc() we start from an empty graph whereas in the second we start from a random graph given the nodes.

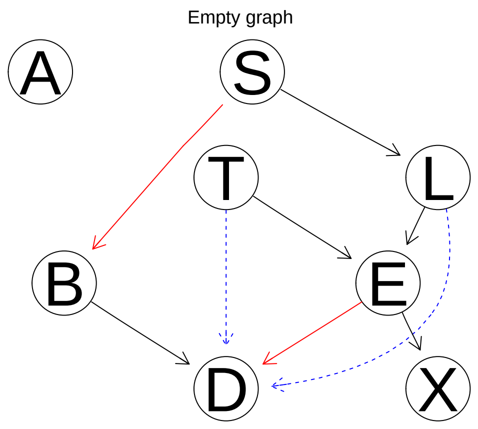
```r
nodes <- colnames(asia)
set.seed(123)
init <- random.graph(nodes)
bn1 <- hc(asia, start = init, score = "bde")
bn2 <- hc(asia, score = "bde")

all.equal(bn1, bn2)
```

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

Below is a plot of the two returned DAGs.

```r
graphviz.compare(bn1, bn2, main = c("Random graph", "Empty graph"))
```





As you can see the two graphs have a different skeleton. Therefore, by definition of equivalence, the two graphs are not equivalent. This is due to the fact that we used two different initializations and we land on two different local minima.

Now, we try different scores without changing the initial graph. The scores that we use are: the multinomial log-likelihood (loglik), the Akaike Information Criterion score (aic) and the default one (Bayesian Information Criterion).

The three runs of the algorithm return three different DAGs. This due to the fact that we are improving different scores.

```
set.seed(123)
bn3 <- hc(asia, score = "aic")
bn4 <- hc(asia, score = "loglik")
bn5 <- hc(asia)
```

```
all.equal(bn3, bn4)
```

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

```
all.equal(bn3, bn5)
```

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

```
all.equal(bn4, bn5)
```

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

Changing the imaginary sample size, the number of restarts and the number of perturbations can also be changed in order to produce non equivalent DAGs.

In conclusion, we showed that by changing the starting point of the HC algorithm or by changing the score (in R this is done by changing the parameters of the hc() function), we land on different local minima.

Question 2

```r
# load the data
data("asia")
# divide data into training and test set
n <- dim(asia)[1]
set.seed(12345)
id <- sample(1:n, floor(n * 0.8))
train <- asia[id, ]
test <- asia[-id, ]
```

We use the Hill Climbing algorithm to learn the structure of the Bayesian network using the training data and the Maximum Likelihood estimates to learn the parameters given the structure.

```r
# learn the graph structure from training data
dag <- hc(train, score = "bde")
```

```r
# learn the graph parameters (local probabilities) from
# training data with maximum likelihood
fitted <- bn.fit(dag, train, method = "mle")

# convert bn.fit object into grain object
fitted_grain <- as.grain(fitted)
# compile the fitted_grain
fitted_grain <- compile(fitted_grain)
```

After learning the graph and the parameters we compute the posterior probability distribution of S (smoking) for each case in the test data set. Then we classify each case in the test data set based on the maximum probability.

The posterior probabilities are computed using exact inference.

```r
# compute posterior probabilities of S=yes
n_test <- dim(test)[1]

probs_yes <- rep(0, n_test)
index_S <- grep("S",names(test))
for (i in 1:n_test) {
  state <- as.character(unlist(test[i, -index_S], use.names = FALSE))
  ev <- setEvidence(fitted_grain, nodes=colnames(asia)[-index_S],
                    state=state)
  prob <- querygrain(ev, nodes = c("S"))$S[2]
  probs_yes[i] <- prob
}
```

```r
# predictions
pred <- ifelse(probs_yes > 0.5, "yes", "no")
```

Confusion matrix estimated model:

```r
# confusion matrix
table(test$S, pred)
```

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

Confusion matrix true model:

```
# true model
dag_true = model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")

# learn the graph parameters (local probabilities) from
# training data with maximum likelihood
fitted <- bn.fit(dag_true, train, method = "mle")
# convert bn.fit object into grain object
fitted_grain <- as.grain(fitted)

# compute posterior probabilities of S=yes
n_test <- dim(test)[1]

probs_yes <- rep(0, n_test)
for (i in 1:n_test) {
  state <- as.character(unlist(test[i, -index_S], use.names = FALSE))
  ev <- setEvidence(fitted_grain, nodes=colnames(asia)[-index_S],
                    state=state)
  prob <- querygrain(ev, nodes = c("S"))$S[2]
  probs_yes[i] <- prob
}

# predictions
pred <- ifelse(probs_yes > 0.5, "yes", "no")

# confusion matrix
table(test$S, pred)
```

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

The results are the same as the DAG estimated with Hill Climbing is equivalent to the one as in the true model.

Question 3

The markov blanket of S is as follows:

```
markovBlanket <- mb(fitted, "S")
markovBlanket
```

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

Repeat exercise above using only markov blanket of S as evidence:

```r
# load the data
data("asia")
# divide data into training and test set
n <- dim(asia)[1]
set.seed(12345)
id <- sample(1:n, floor(n * 0.8))
train <- asia[id, ]
test <- asia[-id, ]

# true DAG
dag_true = model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")

# estimated DAG
dag <- hc(train, score = "bde")

# learn the graph parameters (local probabilities) from
# training data with maximum likelihood
fitted <- bn.fit(dag_true, train, method = "mle")
# convert bn.fit object into grain object
fitted_grain <- as.grain(fitted)

# markov blanket of S

# compute posterior probabilities of S=yes
n_test <- dim(test)[1]

probs_yes <- rep(0, n_test)
for (i in 1:n_test) {
  state <- as.character(unlist(test[i,markovBlanket], use.names = FALSE))
  ev <- setEvidence(fitted_grain, nodes=markovBlanket,
                    state=state)
  prob <- querygrain(ev, nodes = c("S"))$S[2] # p(s|e)
  probs_yes[i] <- prob
}

# predictions
pred <- ifelse(probs_yes > 0.5, "yes", "no")

# confusion matrix
table(test$S, pred)
```

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

As expected the results are the same as when we use all the variables by definition of markov blanket.

Question 4

```r
# create empty BN
naive_DAG <- empty.graph(colnames(asia))
# add arcs according to naive bayes assumption
arc_set <- matrix(c("S", "L", "S", "E", "S", "D",
                    "S", "B", "S", "A", "S", "X", "S", "T"),
             ncol = 2, byrow = TRUE,
             dimnames = list(NULL, c("from", "to")))
arcs(naive_DAG) <- arc_set
```

Repeat exercise above using the Naive Bayes DAG:

```r
# learn the graph parameters (local probabilities) from
# training data with maximum likelihood
fitted <- bn.fit(naive_DAG, train, method = "mle")
# convert bn.fit object into grain object
fitted_grain <- as.grain(fitted)

# compute posterior probabilities of S=yes
n_test <- dim(test)[1]

probs_yes <- rep(0, n_test)
for (i in 1:n_test) {
  state <- as.character(unlist(test[i, -2], use.names = FALSE))
  ev <- setEvidence(fitted_grain, nodes=colnames(asia)[-2],
                   state=state)
  prob <- querygrain(ev, nodes = c("S"))$S[2]
  probs_yes[i] <- prob
}

# predictions
pred <- ifelse(probs_yes > 0.5, "yes", "no")

# confusion matrix
table(test$S, pred)
```

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

The result is slightly worse than the ones above.

Question 5

In question 2 all the results are the same as the graphs are Markov equivalent. In question 3 the result is the same as the results in question 2 since the Markov Blanket of a variable makes the variable independent of the rest of the variables. In this case nodes “B” and “L” cause node “S” to be independent of the rest of the nodes. In other words:

$$P(S|X) = P(S|L, B)$$

with X being the rest of the nodes.

Based on this knowledge, we should expect to get the same result.

However, in question 4 we can see that the dependencies are much simpler (naive), therefore, the accuracy of the inference is lower than all the previous inferences. Moreover, the results are different because the Naive Bayes classifier assumes that given the class, the distributions of the other input variables are independent which is a very strong assumption that is not true in this case.

Appendix

As an appendix we add a method for approximate inference to solve question 2 that could be used instead of exact inference.

```r
library(purrr)
training_idx <- sample(5000, nrow(asia) * 0.8)
asia_training <- asia[training_idx, ]
asia_test <- asia[-training_idx, ]
bn_dag <- hc(asia_training, score = "bde", iss = 2, restart = 13)
param_tables <- bn.fit(bn_dag, asia_training, method = "mle")

target_node_index <- 2
evidence_vector <-
  map_chr(
    list_transpose(as.list(asia_test)),
    ~ paste(
      "(",
      names(asia_test)[-target_node_index],
      " == '",
      map_chr(.x[-target_node_index], as.character),
      "')",
      sep = "",
      collapse = " & "
    )
  )
evnet <- "(S == 'yes')"

cmds <- paste("cpquery(param_tables, ", evnet, ", ",  evidence_vector, ")", sep = "")
probs <- map_dbl( cmds, ~eval(parse(text = .x)) )

y_hats <- ifelse(probs > 0.5, "yes", "no")


table(y_hats, asia_test$S)
```

Statement of Contribution

All the members of the group contributed with code, text and discussions in every question. The individuals solutions of all members were compared in order to decide the best approach. Although the four students contributed to every question a more detailed list of who contributed more to each question would be the following:

- Question 1: Elena Dalla Torre and Arash Haratian
- Question 2: Juan Manuel Pardo Ladino and Elena Dalla Torre
- Question 3: Arash Haratian and Daniel Díaz-Roncero González
- Question 4: Daniel Díaz-Roncero González and Juan Manuel Pardo Ladino
- Question 5: All contributed equally