TDDE15 - Lab1

Oscar Hoffmann, oscho091

September 12, 2024

Q1

Show that multiple runs of the hill-climbing algorithm can return non-equivalent Bayesian network (BN) structures. Explain why this happens.

```r
# Clear the workspace
rm(list = ls())
# Libraries and load the data
library(bnlearn)
library(gRain)
```

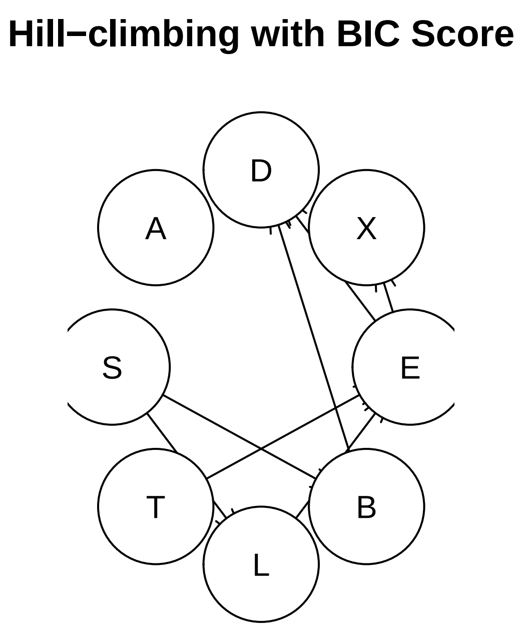
```
## Loading required package: gRbase
```

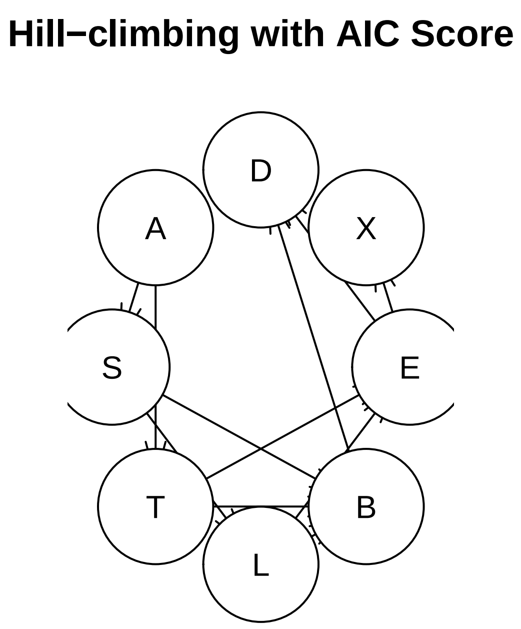
```
##
## Attaching package: 'gRbase'
```

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

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

```r
#Two models, vary score method
model1 = hc(asia, score = "bic")
model2 = hc(asia, score = "aic")

par(mfrow = c(1, 2))
plot(model1, main = "Hill-climbing with BIC Score")
plot(model2, main = "Hill-climbing with AIC Score")
```

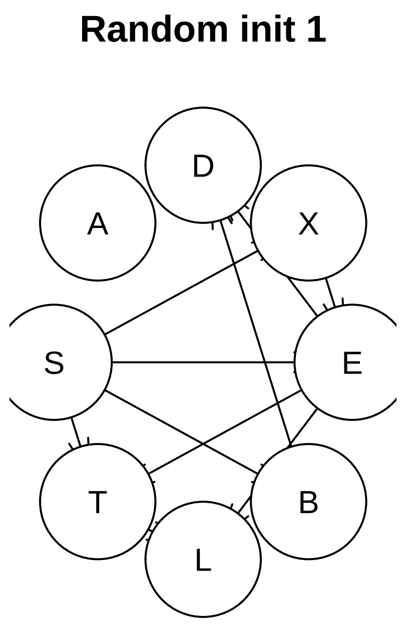


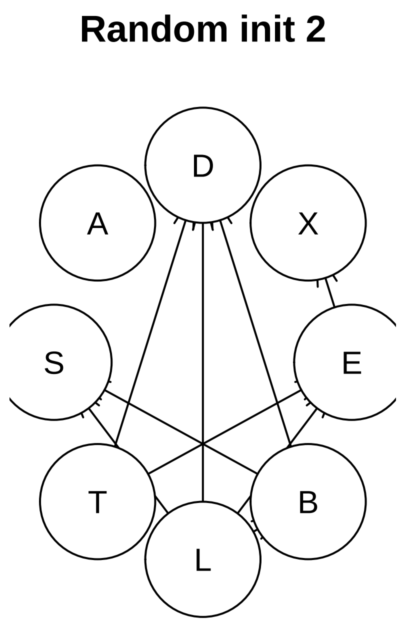


```r
all.equal(cpdag(model1), cpdag(model2))
```

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

```r
#Two models, vary initial structure
set.seed(12345)
model1 = hc(asia, start = random.graph(colnames(asia)))
set.seed(1234)
model2 = hc(asia, start = random.graph(colnames(asia)))

par(mfrow = c(1, 2))
plot(model1, main = "Random init 1")
plot(model2, main = "Random init 2")
```

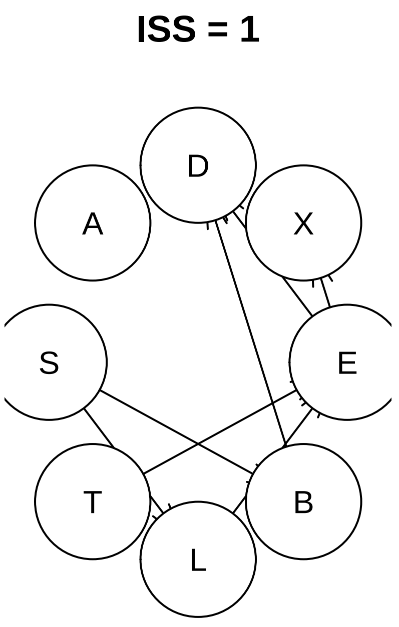


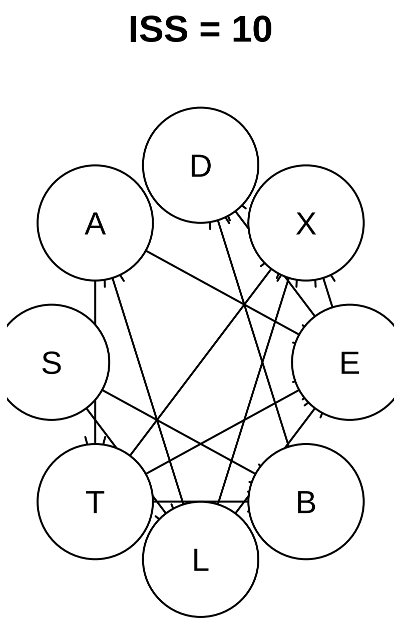


```
all.equal(cpdag(model1), cpdag(model2))
```

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

```
#Two models, vary imaginary sample size
model1 = hc(asia, score = "bde", iss=1)
model2 = hc(asia, score = "bde", iss=10)

par(mfrow = c(1, 2))
plot(model1, main = "ISS = 1")
plot(model2, main = "ISS = 10")
```





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

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

Here I have shown that by varying score method, initial structure and ISS the HC algorithm can yield non-equivalent Bayesian networks. This can been seen in the plots (run the code to see) but also explicitly by converting the models with cpdag and comparing them with all.equals. Cpdag converts the networks to equivalence classes which need to be done to compare them. This is because two Bayesian networks could have different edge directions between nodes, but if the conditional independence properties are the same, they are considered equivalent. However, if they do not encode the same independences, they are non-equivalent. The reason that we end up with non-equivalent BNs is that the HC algorithm is a greedy algorithm that is not guaranteed to find the global optima but rather often get stuck in local optimas.

Q2

Learn a BN from 80% of the Asia dataset. Use the BN learned to classify the remaining 20% of the dataset in two classes: S = yes and S = no. Perform inference and report Confusion Matrix.

```r
rm(list = ls())
data("asia")

#divide into 80% train and 20% test
n=dim(asia)[1]
set.seed(12345)
```

```r
id=sample(1:n, floor(n*0.8))
train=asia[id,]
test=asia[-id,]
```

```r
#learn both structure and init true_dag
model_struct = hc(train, restart = 10)
true_dag = model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")

# learn params
model_params = bn.fit(model_struct, data = train)
true_dag_params = bn.fit(true_dag, data = train)

# Exact inference, transform into gRain objects
model_grain = as.grain(model_params)
model_compiled = compile(model_grain)

true_dag_grain = as.grain(true_dag_params)
true_dag_compiled = compile(true_dag_grain)

# Convert test to a data frame to utilize apply
test = as.data.frame(test)

# Define the nodes (excluding S)
nodes = colnames(test)[-2]

# Function to classify using a given compiled model
classify_with_model <- function(compiled_model, mb_or_S, nodes) {
  #The function here is applied over whole dataframe, 1 means rows, 2 is cols
  apply(test, 1, function(row) {
    # Set evidence for the current row
    evidence <- setEvidence(object = compiled_model,
                            nodes = nodes,
                            states = as.character(unlist(row[mb_or_S]))
                            )

    # Query the posterior probability of S
    query <- querygrain(evidence, nodes = "S")$S

    # Classify based on the probability of "yes"
    ifelse(query["yes"] > 0.5, "yes", "no")
  }
  )
}

# Get predictions for the learned model + cm
pred_model <- classify_with_model(model_compiled, -2, nodes)
table(pred_model, test$S)
```

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

```
# Get predictions for the true DAG model
pred_true_dag <- classify_with_model(true_dag_compiled, -2, nodes)
table(pred_true_dag, test$S)
```

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

The confusion matrices are identical so learnt models is as good at classifing as the true dag

Q3

Classify S given observations only for its Markov blanket. Report the confusion matrix.

```
#Use Markov blanket to do inference more efficiently
mb = mb(model_params, node = "S")
true_mb = mb(true_dag_params, node ="S")

# Get predictions for the learned model with mb
mb_pred <- classify_with_model(model_compiled, mb, mb)
table(mb_pred, test$S)
```

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

```
# Get predictions for the true DAG model with mb
true_mb_pred <- classify_with_model(true_dag_compiled, true_mb, true_mb)
table(true_mb_pred, test$S)
```

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

Using only the Markov blanket variables to classify S yields the same performance as using all other variables. This is expected because the Markov blanket contains all the information needed to predict S.

Q4

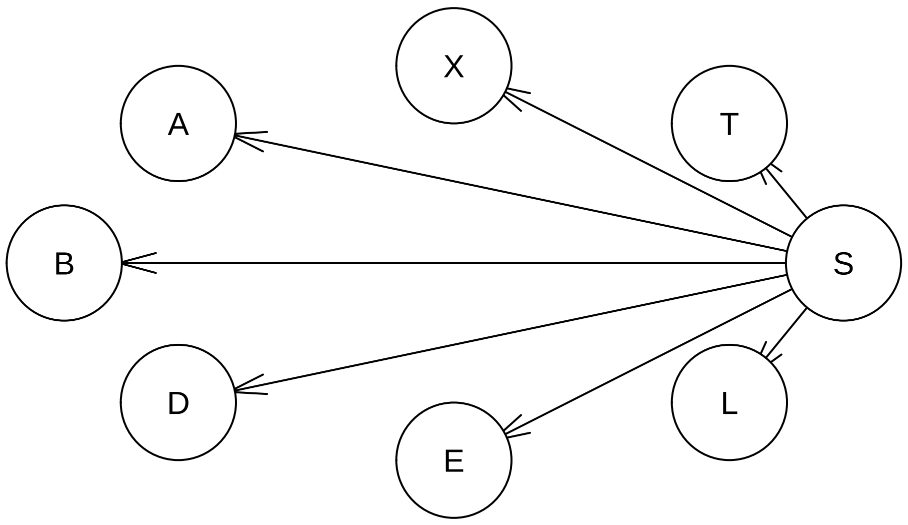
Repeat exercise 2 using a naive Bayes classifier. Model the naive Bayes classifier as a BN created by hand.

```
# Naive Bayes classifier
nb <- model2network("[S][A|S][B|S][D|S][E|S][L|S][T|S][X|S]")
nbc <- bn.fit(x = nb, data = train)

# Transform into gRain obj
nbc <- as.grain(nbc)
```

```r
nbc <- compile(nbc)

# Confusion matrix
table(classify_with_model(nbc, -2, nodes), test$S)
```

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

```r
plot(nb)
```



Q5

Q: Explain why you obtain the same or different results in exercises 2-4.

A: Compared to the confusion matrices from exercises 2, the naive Bayes classifier shows different results, with a decrease in classification performance. The differences arise because the naive Bayes model does not capture the true dependency structure among the variables, leading to less accurate predictions.