Lab 1

Andreas Hertin

andhe794

9/13/2021

Load data

```
data("asia")
## Variables in data:
## A, S, T, L, B, E, X, D

set.seed(10)
```

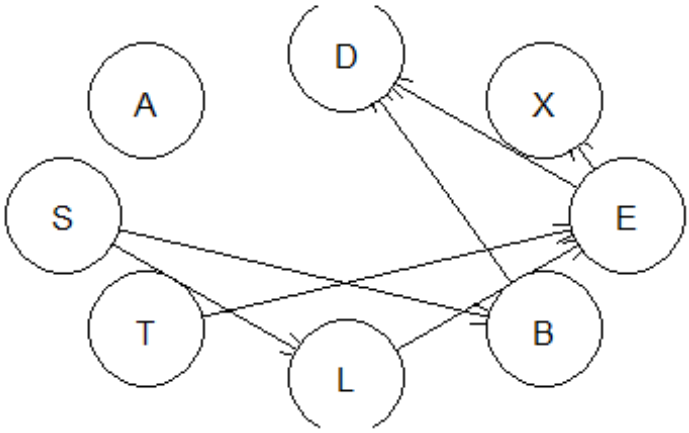
(1) Learn using hill-climb

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'')`. Recall from the lectures that the concept of non-equivalent BN structures has a **precise** meaning.

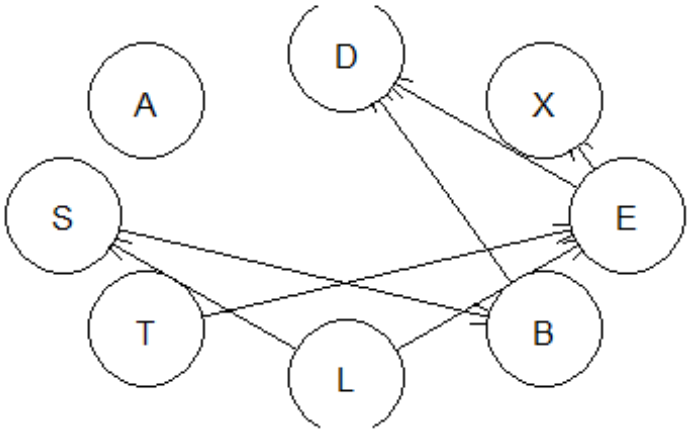
Hint: Check the function `hc()` in the *bnlearn* package. Note that you can specify the initial structure, the number of random restarts, the score, and the equivalent sample size (a.k.a imaginary sample size) in the BDeu score. You may want to use these options to answer the question. You may also want to use the functions `plot(), arcs(), vstructs(), cpdag()` and `all.equal()`.

In the first task I will look at the parameters **start** and **restart** in the `hc()` function to see what difference they make. I did this by comparing results with the functions `plot()`, `all.equal()` and `cpdag()`.

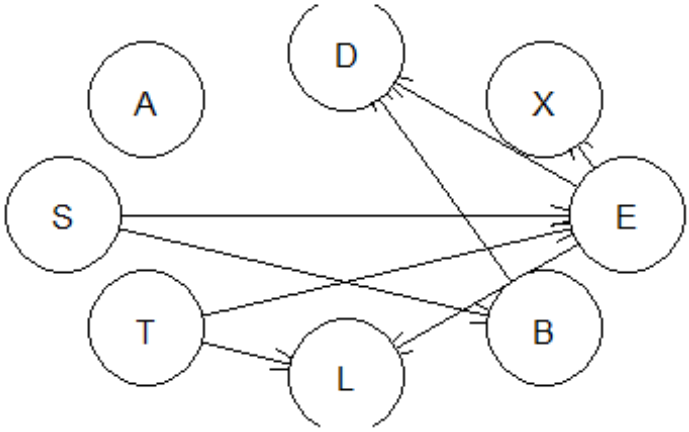
```
HC_0 <- hc(x = asia)

HC_1 <- hc(x = asia,
           restart = 1000)

HC_2 <- hc(x = asia,
           start = random.graph(colnames(asia)))

plot(HC_0)
```




```
plot(HC_1)
```



```
plot(HC_2)
```




```
all.equal(HC_0, HC_1)

## [1] "Different arc sets"

all.equal(HC_0, HC_2)

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

all.equal(cpdag(HC_0), cpdag(HC_1))

## [1] TRUE
```

According to the comparisons the graphs are not in the same equivalence class, in fact they don't even have the same number of arcs. This can also be easily seen in the plots of them, for example the arc T to X is only present in one of the graphs as well as L to X.

The graphs for **HC_0** and **HC_1** are very similar and do infact produce a **TRUE** when comparing their `cpdag()`s. The only different thing is the edge between L and S that changes direction. It is easy to see how different the two graphs of **HC_0** and **HC_2** are by looking at the plots. This stems from **HC_2** having a different random starting network.

(2) Learn both

Learn a BN from 80 % 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

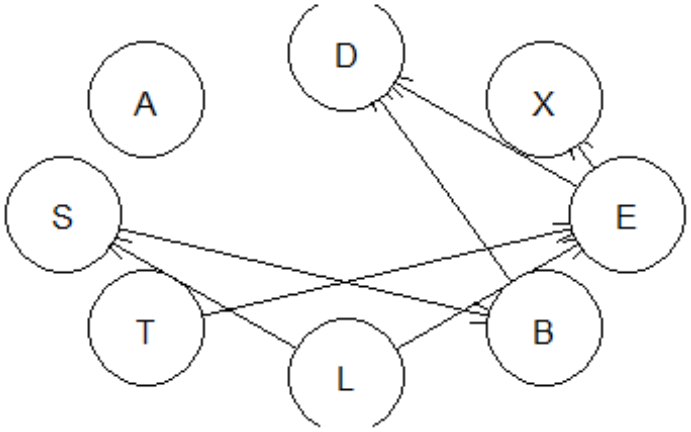
learning algorithm and settings that you consider appropriate. Use the BN learned to classify the remaining 20 % of the Asia dataset in two classes: *S = yes* and *S = no*. In other words, compute the posterior probability distribution of S for each case and classify it in the most likely class. To do 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]")`.

Hint: You already know two algorithms for exact inference in BNs: Variable elimination and cluster trees. There are also approximate algorithms for when the exact ones are too demanding computationally. For exact inference, you may need the functions `bn.fit()` and `as.grain()` from the *bnlearn* package, and the functions `compile()`, `setEvidence()` and `querygrain` from the package *gRain.* For approximate inference, you may need the functions `cpquery()` or `cpdist()`, `prop.table()` and `table()` from the *bnlearn* package

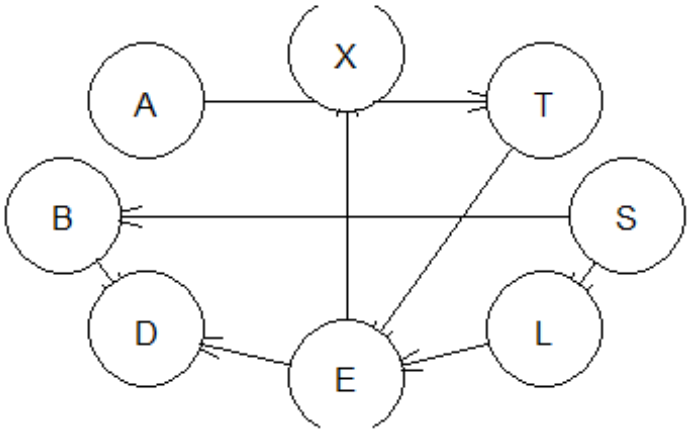
For this task I created a function to get the predictions from the dataset when fitted with `bn.fit()` from the *bnlearn* package. This makes it so that I can compare the confusion matrices of the real DAG and the learned DAG.

```
predictNet <- function(juncTree, data, features, target){
  predArray <- matrix(nrow = nrow(data),
                      ncol = 1)

  for (i in 1:nrow(data)) {
    obsStates <- NULL
    for (j in features) {
      obsStates[j] <- ifelse(data[i, j] == "yes", "yes", "no")
    }

    obsEvidence <- setEvidence(object = juncTree,
                               nodes = features,
                               states = obsStates)

    obsPredProb <- querygrain(object = obsEvidence,
                              nodes = target)$S

    predArray[i] <- ifelse(obsPredProb["yes"] >= 0.5, "yes", "no")
  }

  return(predArray)

}


trainingIndices = sample(1:dim(asia)[1],
                         floor(dim(asia)[1] * 0.8))
```

```
asia.train <- asia[trainingIndices,]
asia.test <- asia[-trainingIndices,]

set.seed(1)
HC_BN_test <- hc(x = asia.train,
                 restart = 100)

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

plot(HC_BN_test)
```




```
plot(HC_BN_true)
```




```
fitTest <- bn.fit(HC_BN_test,
                  asia.train)

fitTrue <- bn.fit(HC_BN_true,
                  asia.train)

grainTest <- as.grain(fitTest)
grainTrue <- as.grain(fitTrue)

fitTestJunctionTree <- compile(grainTest)
fitTrueJunctionTree <- compile(grainTrue)

observedVars = c("A", "T", "L", "B", "E", "X", "D")
targetVar    = c("S")

predTest <- predictNet(fitTestJunctionTree,
                       asia.test,
                       observedVars,
                       targetVar)

predTrue <- predictNet(fitTestJunctionTree,
                       asia.test,
                       observedVars,
                       targetVar)
```

```
confMatrixTest <- table(predTest,
                                  asia.test$S)

confMatrixTrue <- table(predTrue,
                                  asia.test$S)

print(confMatrixTest)
```

```
##
## predTest  no yes
##      no  340 120
##      yes 157 383
```

```
print(confMatrixTrue)
```

```
##
## predTrue  no yes
##      no  340 120
##      yes 157 383
```

We can see from the confusion matrices that they are exactly the same. The DAG that is output from the `hc()` algorithm is the same as the real one when looking at the target value *S*. They both have the same markov blanket.

Only differnce in the graphs are that the real graph has an edge from A to T.

(3) Markov blanket

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.

Hint: You may want to use the function `mb()` from the *bnlearn* package

```
mbTrue <- mb(fitTrue,
             node = "S")


predMbTrue <- predictNet(fitTrueJunctionTree,
                         asia.test,
                         mbTrue,
                         targetVar)

confMatrixMbTrue <- table(predMbTrue,
                          asia.test$S)
```

```
print(confMatrixMbTrue)
```

```
##
## predMbTrue  no yes
##        no  340 120
##        yes 157 383
```

The confusion matrix is the same as in **(2)** which is as expected since the markov blanket is once again the same.

(4) Naive BN

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.

Hint: Check *http://www.bnlearn.com/examples/dag/* to see how to create a BN by hand.

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

fitNaive <- bn.fit(naiveBN,
                   asia.test)

grainNaive <- as.grain(fitNaive)

junctionNaive <- compile(grainNaive)

predNaive <- predictNet(junctionNaive,
                        asia.test,
                        observedVars,
                        targetVar)

confMatrixNaive <- table(predNaive,
                         asia.test$S)

print(confMatrixNaive)
```

```
##
## predNaive  no yes
##       no  364 189
##       yes 133 314
```

Now we got a different, slightly worse, confusion matrix. We can conclude that there is a different markov blanket that contains all variables instead of only *B* and *L* as before.

5

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

As said in previous explanations the result varies depending on the markov blanket of S from the models.