Lab 1 report – William Anzén TDDE15

Assignment 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").

When trying different starting models for the HC-algorithm the following was found:

"Different number of directed/undirected arcs"

Conclusion: Different local optimums were found while using different starting models.

This makes sense since the algorithm, through iteration, will end up in a different solution.

When trying different restart values, the following was found:

Different arc sets found when restart = 105, 1000.

Conclusion: There is some randomness to the HC-algorithm, but quite rarely does it have an impact.

Choosing a higher restart-value will often result in a network with higher score.

When changing the scoring methods, the following was found:

No difference, i.e only changing the scoring method did not result in a different network.

Conclusion: The different scoring methods are still resulting in the highest scoring network.

When trying different imaginary sample sizes, the following was found:

"Different number of directed/undirected arcs"

Conclusion: ISS determines the regularization applied when creating the network with the algorithm. A high value for ISS means less regularization, which ultimately ends up in more complex solutions with more edges in the network. Meanwhile a lower value for ISS means more regularization, resulting in a more sparse solution with less edges.

Tried the other functions – see code in the appendix.

Assignment 2

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]").

Test stands for my own network structure with my own inputs. Tried to make this network differ from the original network structure.

True stands for the original.

```
> confusionTest

predictionTest  no yes
           no  337 121
           yes 176 366
```

```
> confusionTrue

predictionTrue  no yes
           no  337 121
           yes 176 366
```

The results were no difference in the predictions.

Assignment 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.

```
> markovConfusionTrue

mbPredTrue  no yes
       no  337 121
       yes 176 366
```

The confusion matrix is the same as in assignment 2.

Assignment 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.

```
> confusionNaive

predNaive  no yes
       no  359 180
       yes 154 307
```

We build the network with all nodes depending on the class variable S. The result is a different confusion matrix from the previous assignments.

Assignment 5

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

Markov Blanket (MB) consists of the nodes of importance for the target nodes' dependencies in the network. If we do not change the MB, the outcome of the prediction on the target node will not change. Since the MB does not change from (2) to (3) nor the two different bayesian networks used in task (2), therefore the result does not change between these assignments. When we perform the naive bayes network, the MB has drastically changed from containing {B, L} to containing all nodes. Therefore, we get a different result in (4).

Appendix CODE

```
# if (!requireNamespace("BiocManager", quietly = TRUE))

#   install.packages("BiocManager")

# BiocManager::install("RBGL")

# BiocManager::install("Rgraphviz")

# BiocManager::install("gRain")
```

```
#
# install.packages("bnlearn")
# library(bnlearn)
# library(RBGL)
# library(Rgraphviz)
# library(gRain)

# (1)

data("asia")
hcOne <- hc(asia) #Standard HC-algorithm - comparing from this network
hcTwo <- hc(asia, start = model2network("[B][T][A|T][S|A][L|A][D|B:L][E|T:L][X|E]"))



all.equal(hcOne,hcTwo)
plot(hcTwo)
plot(model2network("[B][T][A|T][S|A][L|A][D|B:L][E|T:L][X|E]"))
#"Different number of directed/undirected arcs"
#Local optimum found while using a different start model.
#Makes sense since the algorithm through iteration will end up in another solution.

hcThree <- hc(asia, restart = 1000)
all.equal(hcOne,hcThree)
#Different arc sets found when restart = 105, 1000, otherwise no difference.
#Conclusion, there is some randomness to the HC algorithm but quite rarely does it have an impact.
#Choosing a higher restart-value will often result in a network with higher score.

hcFour <- hc(asia, score = "bde")
all.equal(hcOne,hcFour)
```

```
#True, i.e only changing the scoring method did not result in a different network.
#Conclusion the scoring methods are still resulting in the highest scoring network.

hcFive <- hc(asia, score="bde", iss=5)
all.equal(hcOne,hcFive)
#"Different number of directed/undirected arcs" iss determines the weight of the prior,
# here iss=5 compared to iss=1 which results in a different local optimum

vstructs(hcFive) #Z-Child X,Y - Parents
vstructs(hcOne)
plot(hcFive)

cpdag(hcOne) #Summary
cpdag(hcFive)
arcs(hcOne) #Printing the arcs

#(2)



#Function to compute the conditional probability of each observation
predictNet <- function(juncTree, data, features, target){
  predArray <- matrix(nrow=nrow(data),ncol=1)

  for(i in 1:nrow(data)){
    obsStates <- NULL
    for(p in features){
      if(data[i,p]=="yes"){
        obsStates <- c(obsStates,"yes")
      } else{
```

```
      obsStates <- c(obsStates,"no")
    }
  }


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

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

  predArray[i] <- if(obsPredProb["yes"]>=0.5) "yes" else "no"

 }
 return(predArray)
}



n=dim(asia)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.8))
train=asia[id,]
test=asia[-id,]

hcBNTest <- hc(train, start = model2network("[B][T][A|T][S|A][L|A][D|B:L][E|T:L][X|E]"), score="bde",
iss=5)
BNTrue <- model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")
plot(hcBNTest)
```

```
plot(BNTrue)

fittedTest <- bn.fit(hcBNTest, train) #Fitting parameters in structure with traindata

fittedTrue <- bn.fit(BNTrue, train)

BNGrainTest = as.grain(fittedTest) #Graphical independence network

BNGrainTrue = as.grain(fittedTrue)

junctionTreeTest <- compile(BNGrainTest) #Get cliques with potentials (Lauritzen-Spiegelhalter
algorithm

predictionTrue <- predictNet(junctionTreeTrue, test, obsVars, tarVar)


confusionTest <- table(predictionTest, test$S)

confusionTest

confusionTrue <- table(predictionTrue, test$S)

confusionTrue

#Same confusion tables


# (3)



markovObsVarsTrue <- mb(BNTrue, "S")

mbPredTrue <- predictNet(junctionTreeTrue, test, markovObsVarsTrue, tarVar)


markovConfusionTrue <- table(mbPredTrue, test$S)

markovConfusionTrue

#Same confusion table as in (2)


# (4)


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

fittedNaive <- bn.fit(naiveBayesNet, train)
```

```
naiveGrain <- as.grain(fittedNaive)

naiveJunction <- compile(naiveGrain)

mb(naiveBayesNet, "S")


predNaive <- predictNet(naiveJunction,test,obsVars,tarVar)


confusionNaive <- table(predNaive, test$S)

confusionNaive

confusionTrue


# (5)

# Markov Blanket (MB) consists of the nodes of importance for the target nodes' dependencies in the network.

# If we do not change the MB, the outcome of the prediction on the target node will not change.

# Since the MB does not change from (2) to (3) nor the two different bayesian networks used in

# task (2), therefore the result does not change between these assignments.

# When we perform the naive bayes network, the MB has drastically changed from containing B,L to containing

# all the other nodes. Therefore we get a different result in (4).
```