TDDE15 - Lab 1

Erik Jareman

2022-09-08

```
set.seed(1)
data("asia")
```

Task 1

```
# Learn Bayesian networks using different score functions
bn1 = hc(asia, score="bde", iss=3, restart=5)
bn2 = hc(asia, score="loglik", restart=5)

# Check if equivalence classes of networks are equal
all.equal(cpdag(bn1), cpdag(bn2))
```

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

Since the equivalence classes are not equal, we know that the BN structures are non-equivalent. This is because the hill climbing algorithm has found different local maximums for the two networks. This could be due to the low number of random restarts used (5), but it is more likely due to the fact that we are using different score functions with different local maximums.

Task 2

The hill climbing algorithm is used to learn a Bayesian network based on the training set. This network (with fitted parameters) is then used to classify the samples in the testing set using exact inference with junction trees. The same method is then used to classify the same samples using the true network structure given in the lab.

```
# Split asia data into training and testing set
train80 = sort(sample(nrow(asia), nrow(asia)*.8))
asiaTrain = asia[train80, ]
asiaTestLabeled = asia[-train80, ]
asiaTest = asiaTestLabeled[,-2] # Remove col 'S' from testing set

# Use hill climbing algorithm to learn the structure
learnedBN = hc(x=asiaTrain, score="bde", restart=100)

# Fit the parameters conditional on the network structure (MLE)
fit_bnlearn = bn.fit(x=learnedBN, data=asiaTrain)
```

```
# use exact inference with junction trees (gRain package)
fit_gRain = as.grain(fit_bnlearn)
juncTree = compile(fit_gRain)

# Classify each sample in asiaTest
targetNode = c("S")
observedNodes = names(unlist(asiaTest[1, ]))
classification = c()

for (sampleRow in 1:nrow(asiaTest))
{

  evidence = setEvidence(object=juncTree,
                         nodes=observedNodes,
                         states=as.character(unlist(asiaTest[sampleRow, ])))

  pred = querygrain(object=evidence,
                    nodes=targetNode)

  classification[sampleRow] = names(which.max(pred$S))

}

# Predict all samples in the test-set, using the true BN
trueBN = model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")
trueFit_bnlearn = bn.fit(x=trueBN, asiaTrain)
trueFit_grain = as.grain(trueFit_bnlearn)
trueJuncTree = compile(trueFit_grain)

trueClassification = c()
for (sampleRow in 1:nrow(asiaTest))
{
  evidence = setEvidence(object=trueJuncTree,
                         nodes=observedNodes,
                         states=as.character(unlist(asiaTest[sampleRow, ])))
  pred = querygrain(object=evidence,
                    nodes=targetNode)
  trueClassification[sampleRow] = names(which.max(pred$S))
}
```

Task 3

Classify the samples of the testing data using only the markov blanket of S.

```
# get the markov blanket of S given the Bayesian parameters learned
markovBlanket = mb(x=fit_bnlearn,
                   node=targetNode)

# predict S using only observations in markov blanket (nodes "B" and "L")
mbClassification = c()
for (sampleRow in 1:nrow(asiaTest))
{
  evidence = setEvidence(object=juncTree,
```

```
                          nodes=markovBlanket,
                          states=as.character(c(asiaTest[sampleRow, ]$B,
                                                asiaTest[sampleRow, ]$L)))

  pred = querygrain(object=evidence,
                    nodes=targetNode)

  mbClassification[sampleRow] = names(which.max(pred$S))
}
```

Task 4

Classify the samples in the testing data using a naive Bayes approach.

```
# Set the naive Bayes structure given S (all variables are independent given S)
# Predict as done in Task 2 and 3
naiveBN = model2network("[S][A|S][T|S][L|S][B|S][E|S][X|S][D|S]")
naiveFit_bnlearn = bn.fit(x=naiveBN, data=asiaTrain)
naiveFit_gRain = as.grain(naiveFit_bnlearn)
naiveJuncTree = compile(naiveFit_gRain)

naiveClassification = c()
for (sampleRow in 1:nrow(asiaTest))
{
  evidence = setEvidence(object=naiveJuncTree,
                          nodes=observedNodes,
                          states=as.character(unlist(asiaTest[sampleRow, ])))

  pred = querygrain(object=evidence,
                    nodes=targetNode)

  naiveClassification[sampleRow] = names(which.max(pred$S))
}
```

Task 5

Calculate the confusion matrix and the accuracy for the learned model, true model, markov blanket model, and the naive bayes model.

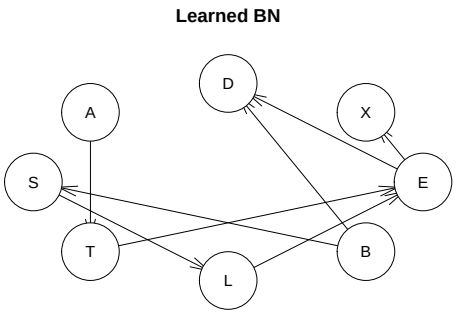
```
trueCM = table(trueClassification, asiaTestLabeled$S)
trueAcc = sum(diag(trueCM))/sum(trueCM)

learnedCM = table(classification, asiaTestLabeled$S)
learnedAcc = sum(diag(learnedCM))/sum(learnedCM)

markovBlanketCM = table(mbClassification, asiaTestLabeled$S)
markovBlanketAcc = sum(diag(markovBlanketCM))/sum(markovBlanketCM)

naiveBayesCM = table(naiveClassification, asiaTestLabeled$S)
naiveBayesAcc = sum(diag(naiveBayesCM))/sum(naiveBayesCM)
```

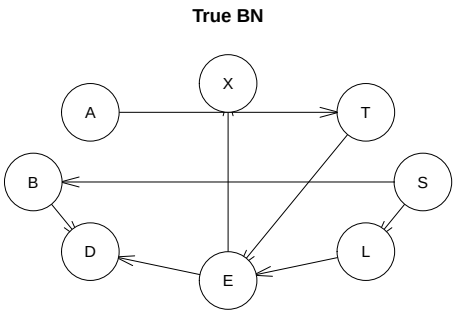
trueAcc: 0.723
learnedAcc: 0.723
markovBlanketAcc: 0.723
naiveBayesAcc: 0.694

The same accuracy (and same classification for each sample) is obtained when using the true BN, the learned BN, and the markov blanket BN. The accuracy when using the naive Bayes approach is slightly lower.

In task 2, the network structure of the learnt network is very similar to the true network structure. Therefore, it is reasonable that these networks classify the samples in a similar manner. The hill climbing algorithm with score function bde did a good job when learning the network structure.

```
plot(learnedBN, main="Learned BN")
plot(trueBN, main="True BN")
```





In task 3, only the observations in the markov blanket are used to classify the samples. It is reasonable for this network to give the same classification as before, since the markov blanket makes S independent of all other variables. (The networks in task 2 should have the same markov blanket as the network in task 3, and the markov blanket is enough to predict S. Therefore, the result should be the same.)

In task 4, the naive Bayes approach is used, and the network should therefore be simplified compared to the networks in the earlier tasks. For this reason, it is expected for this network to classify the samples with less accuracy than in the earlier tasks. And that is exactly what the results show.