Lab Report in Advanced Machine Learning

Laboration 1

TDDE15

David Grönberg
Davgr686

11-09-2020

Contents

List of Figures

1. Assignments

1.1 Assignment 1

To show that multiple runs of the hill-climbing function can return non-equivalent Bayesian network structures, the function *hc* is used to generate two Bayesian network structures using different hyperparameters. The dataset used to learn the structures is the Asia dataset. First, the Bayesian network structure is learned using the Bayesian Information Criterion score, with a random starting DAG. Next, the second Bayesian network structure is learned using the same score, but with a new random starting DAG. The two networks graphs are plotted and compared.

Figures 1.1 and 1.2 show the two networks graph structures, where one can clearly see that they do not share the same structure. The difference between the two structures can also be confirmed by running the function *all.equal*() which outputs "Different number of directed/undirected arcs". This can be explained by that the hill-climbing algorithm only finds local optima (when the problem is not convex), which is not always the global optima. Thus, if the starting points are different, the runs might find different local optima, resulting in different structures.

```r
data(asia)

bnet1 <- hc(x = asia, start = random.graph(names(asia)), score = "bic")
bnet1 <- cpdag(bnet1)

bnet2 <- hc(x = asia, start = random.graph(names(asia)), score = "bic")
bnet2 <- cpdag(bnet2)

all.equal(bnet1, bnet2)
#OUTPUT: "Different number of directed/undirected arcs"
```

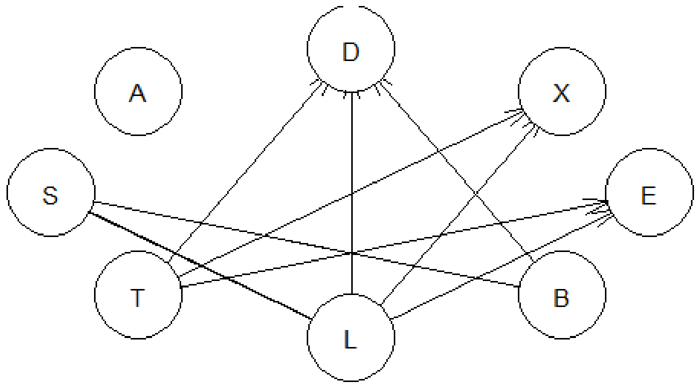

Figure 1.1: Graph of the first learned Bayesian Network using hill-climbing

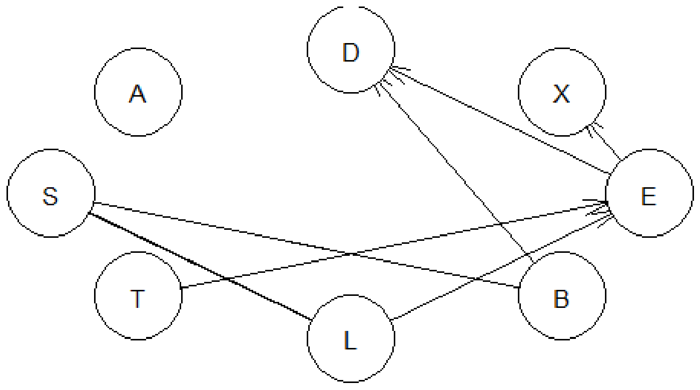

Figure 1.2: Graph of the first learned Bayesian Network using hill-climbing

1.2 Assignment 2

The next task is to learn a Bayesian network using 80% of the *Asia* dataset and then use the learned network to classify the *smoke (S)* variable in the remaining 20% of the dataset.

The hill-climbing algorithm is used to learn the structure of the network, using the *Bayesian Information Criterion* score. The learned Bayesian network is shown in figure 1.3. The parameters are fit using the *bn.fit()* function.

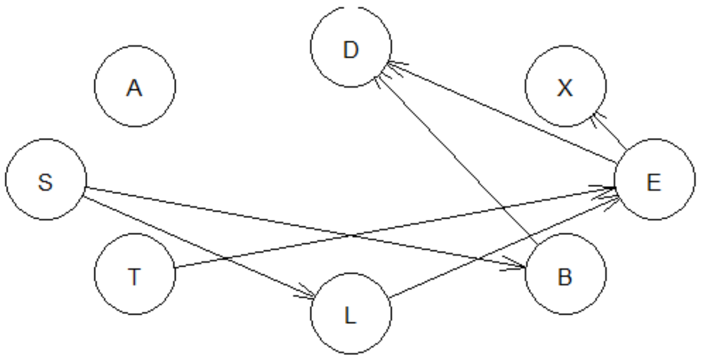

Figure 1.3: Graph of the learned Bayesian Network

The Bayesian network is used to classify the S variable by iterating the test set, and for each iteration, calculating the conditional probability of $S = "yes"$ conditioned on the observations of the other variables. If the conditional probability of $S = "yes" >= 0.5$, we classify S as "yes", otherwise "no". The conditional probability is obtained using the *cpquery()* function and the approximate inference algorithm Likelihood weighting.

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

predict_smoke_asia <- function(fitted, data) {
  preds <- c()
  for (i in 1:dim(data)[1]) {
    sample <- unlist(data[i, -2])
    prob <- cpquery(fitted,
                    event=eval(parse(text="(S=='yes')")),
                    evidence=list(A=sample[1], T = sample[2], L = sample[3], B = sample[4], E
    = sample[5], X = sample[6], D = sample[7]),
                    method="lw")
    if (prob >= 0.5) {
      preds[i] <- 'yes'
    }
    else {
      preds[i] <- 'no'
    }
  }
  return (preds)
}

bnet <- hc(x = train, restart = 5, score = "bic")
bnet_fitted <- bn.fit(bnet, train)
plot(bnet)
preds <- predict_smoke_asia(bnet_fitted, test)
mean(preds == test$S)
prop.table(table("preds" = preds, "true" = test$S))

dag = model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")
bnet_true_fitted <- bn.fit(dag, train)
plot(dag)
preds <- predict_smoke_asia(bnet_true_fitted, test)
mean(preds == test$S)
prop.table(table("preds" = preds, "true" = test$S))
```

The true Asia Bayesian network structure is also fit and used to predict the S variable in test set. The results of the predictions from the two Bayesian networks are compared. The tables show that the resulting confusion matrix for the different Bayesian networks are identical. This can be explained by that the networks share the same markov blanket for S.

Table 1.1: Confusion matrix for the predictions derived from the fitted Bayesian network

		Predictions	
		No	Yes
True	No	0.350	0.148
	Yes	0.104	0.398

Table 1.2: Confusion matrix for the predictions derived from the true Asia Bayesian network

		Predictions	
		No	Yes
True	No	0.350	0.148
	Yes	0.104	0.398

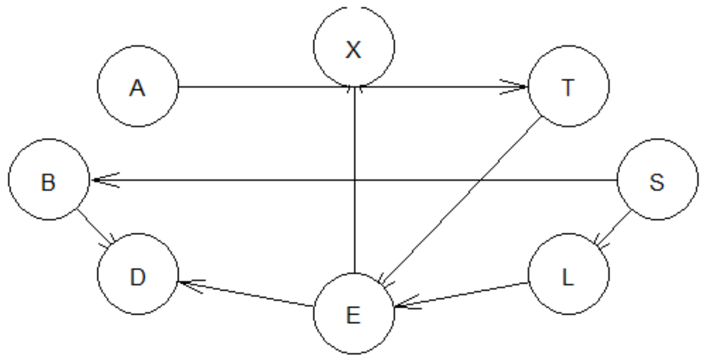

Figure 1.4: Graph of the true Asian Bayesian Network

1.3 Assignment 3

The next task is to predict the same S variable using only observations from the markov blanket of S. The markov blanket for S contains the variables L & B, for both networks. The confusion matrix is reported.

The predictions using only the markov blanket of S yields the same confusion matrix as to when using all the variables. This can be explained by that all the other variables are independent of S, given observations from L and B (chain rule).

Table 1.3: Confusion matrix for the predictions derived from the fitted Bayesian network using the markov blanket

		Predictions	
		No	Yes
True	No	0.350	0.148
	Yes	0.104	0.398

Table 1.4: Confusion matrix for the predictions derived from the true Asia Bayesian network using the markov blanket

		Predictions	
		No	Yes
True	No	0.350	0.148
	Yes	0.104	0.398

1.4 Assignment 4

Lastly, the prediction of S is repeated using a naive bayes bayesian network. The network is constructed by making the S variable the parent to all the other variables, resulting in the structure shown in figure 1.5. Thus, all the predictive varaibles are conditionally independent of each other given S. The resulting confusion matrix from the predictions are shown below.

Table 1.5: Confusion matrix for the predictions derived from the naive bayes Bayesian network

		Predictions	
		No	Yes
True	No	0.382	0.132
	Yes	0.162	0.324

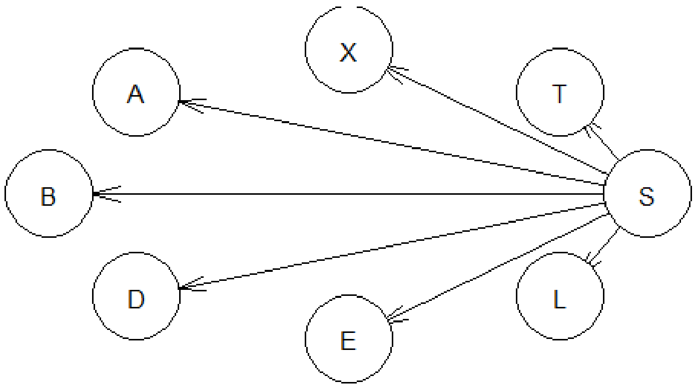

Figure 1.5: Graph of the naive bayes Bayesian Network

1.5 Assignment 5

The predictions in assignment 2 and 3 resulted in the same confusion matrix. This can be explained by that all the variables that are not in the Markov blanket of S are independent of S, given observations from L and B (chain rule). Thus, leading to the same conditional probability for S in both setups when L and B are observed.

The accuracy of the predictions of S when using the naive Bayes Bayesian network was slightly lower compared to when using the networks in assignment 2 and 3. The naive Bayes Bayesian network makes the naive assumption that all the predictive variables are independent of each other. i.e. that L and B are independent of each other, which we can see that they are not when inspecting the structure of the true Asia Bayesian network.

2. Code Appendix

2.1 Assignment 1

```r
library(bnlearn)
set.seed(1234)

#Note that hill-climbing always returns a DAG, not a CPDAG; so the
#correct way of comparing it with another graph is to take the CPDAG
#for both.
bnet1 <- hc(x = asia, start = random.graph(names(asia)) , score = "bic")
bnet1 <- cpdag(bnet1)
plot(bnet1)

bnet2 <- hc(x = asia, start = random.graph(names(asia)), score = "bic")
bnet2 <- cpdag(bnet2)
plot(bnet2)
all.equal(bnet1, bnet2)
# "Different number of directed/undirected arcs"
hamming(bnet1, bnet2)
## hamming distance return 6. zero means a perfect match
```

2.2 Assignment 2

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

predict_smoke_asia <- function(fitted, data) {
  preds <- c()
  for (i in 1:dim(data)[1]) {
    sample <- unlist(data[i, -2])
    prob <- cpquery(fitted,
                    event=eval(parse(text="(S=='yes')")),
                    evidence=list(A=sample[1],
                                  T = sample[2],
                                  L = sample[3],
                                  B = sample[4],
                                  E = sample[5],
                                  X = sample[6],
                                  D = sample[7]),
                    method="lw")
    if (prob >= 0.5) {
      preds[i] <- 'yes'
    }
    else {
      preds[i] <- 'no'
    }
  }
  return (preds)
}



bnet <- hc(x = train, score = "bic")
bnet_fitted <- bn.fit(bnet, train)
plot(bnet)
preds <- predict_smoke_asia(bnet_fitted, test)
mean(preds == test$S)
prop.table(table("true" = test$S, "preds" = preds))


dag = model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")
bnet_true_fitted <- bn.fit(dag, train)
plot(dag)
preds_true <- predict_smoke_asia(bnet_true_fitted, test)
mean(preds_true == test$S)
prop.table(table("true" = test$S, "preds" = preds_true))
```

2.3 Assignment 3

```
predict_smoke_asia_mb <- function(fitted, data) {
  preds <- c()
  for (i in 1:dim(data)[1]) {
    sample <- unlist(data[i, -2])
    prob <- cpquery(fitted,
                    event=eval(parse(text="(S=='yes')")),
                    evidence=list(L = sample[3], B = sample[4]),
                    method="lw")
    if (prob >= 0.5) {
      preds[i] <- 'yes'
    }
    else {
      preds[i] <- 'no'
    }
  }
  return (preds)
}

preds <- predict_smoke_asia_mb(bnet_fitted, test)
mean(preds == test$S)
prop.table(table("true" = test$S, "preds" = preds))


preds <- predict_smoke_asia_mb(bnet_true_fitted, test)
mean(preds == test$S)
prop.table(table("true" = test$S, "preds" = preds))
```

2.4 Assignment 4

```
## Xi, Xj independent | S, for all Xi,Xj
## P(s,X1...Xn) = P(S)P(X1|S)*P(X2|S)...P(Xn|S)
n_bn <- model2network("[S][A|S][T|S][L|S][B|S][E|S][X|S][D|S]")
n_bn_fitted <- bn.fit(n_bn, train)
plot(n_bn)
preds <- predict_smoke_asia(n_bn_fitted, test)
mean(preds == test$S)
```

```
prop.table(table("true" = test$S, "preds" = preds))
```