Sol_lab1

Isak Berntsson

9/5/2022

```
library(bnlearn)
```

```
## Warning: package 'bnlearn' was built under R version 4.1.3
```

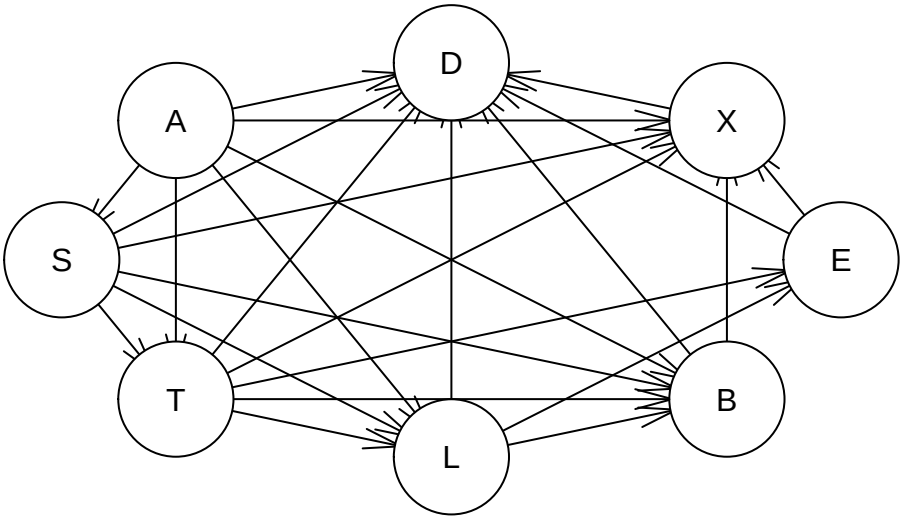
```
data("asia")
dat = asia

head(dat)
```

```
##    A   S   T  L   B   E   X   D
## 1 no yes  no no yes  no  no yes
## 2 no yes  no no  no  no  no  no
## 3 no  no yes no  no yes yes yes
## 4 no  no  no no yes  no  no yes
## 5 no  no  no no  no  no  no yes
## 6 no yes  no no  no  no  no yes
```

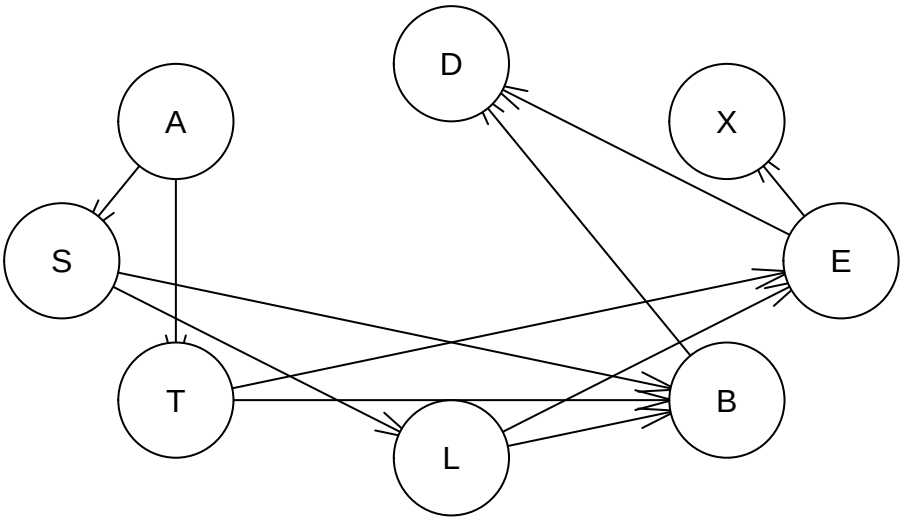
question 1

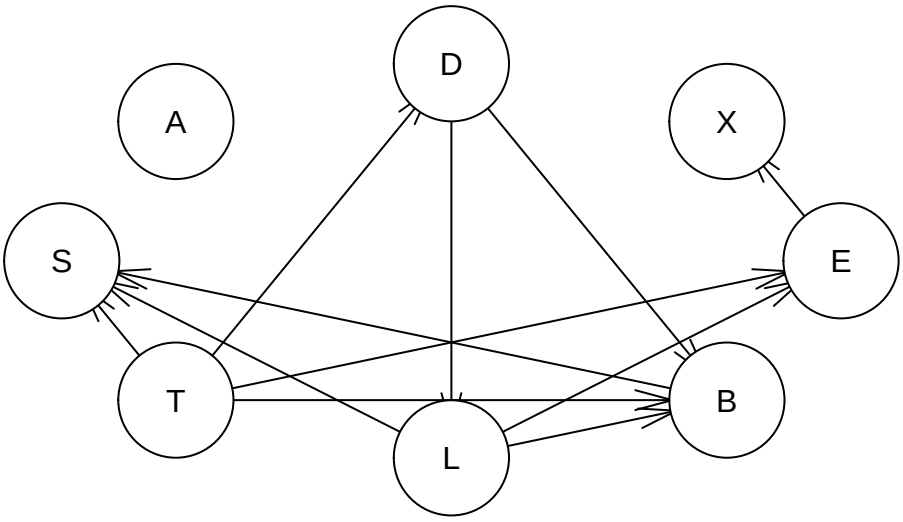
```
set.seed(123)
for (sc in c("loglik", "aic", "k2")){

  mod_base = hc(dat, score=sc)
plot(mod_base, main=paste("scoring function:",sc))
}
```

scoring function: loglik



scoring function: aic





the learned networks all have non-equivalent structures. Suggesting that a different scoring function yields different best networks.

Question 2

Learning and fitting a network from 80% of the data.

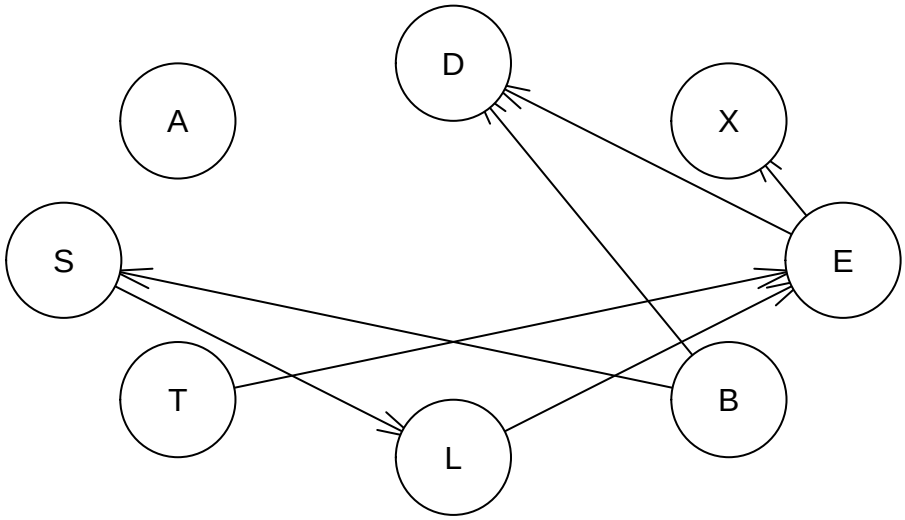
```
set.seed(123)
N = nrow(dat)
fit_num = N*.8

fit_ind = sample(1:N,size = fit_num )

train_dat = dat[fit_ind,] #80%,
test_dat = dat[-fit_ind,]# 20%

print(mean(train_dat$S=="yes"))
```

```
## [1] 0.50375
```

```
print(mean(test_dat$S=="yes"))
```

```
## [1] 0.5
```

```
#learning model and paramets from training_set
mod_80 = hc(train_dat,score="bic", restart=5 )
fitted = bn.fit(mod_80,train_dat)

plot(mod_80)
```



Approximate inference using cpquery()

```
set.seed(123)
results = matrix(nrow=nrow(test_dat),ncol=1)
for (i in 1:nrow(test_dat)){
  row = as.matrix(test_dat[i,])
  prob  = cpquery(fitted,
          event = S=="yes",
          evidence = (A==test_dat[i,1]) & (T==test_dat[i,3]) & (L==test_dat[i,4]) & (B==test_dat[i,5]) &
          )

  results[i] = prob
  }

#
conf_matrix = table(results>=.5,test_dat$S)
accuracy = (conf_matrix[1,1]  + conf_matrix[2,2]) / sum(conf_matrix)
print(conf_matrix)
```

```
##
```

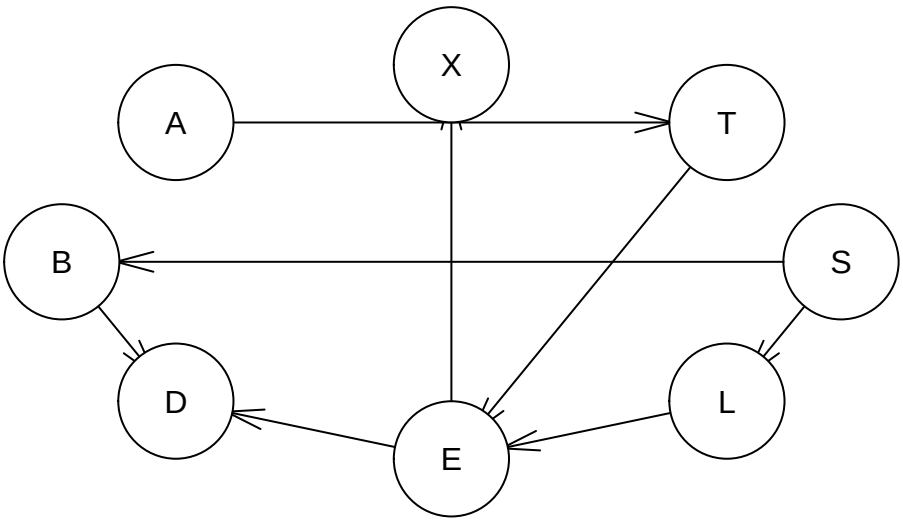
```
##          no yes
##   FALSE 359 119
##   TRUE  141 381
```

```
print(paste("accuracy:",accuracy))
```

```
## [1] "accuracy: 0.74"
```

Loading, and fitting true asia BN

```
true_net =model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")

plot(true_net)
```




```
true_fit = bn.fit(x =true_net , train_dat)
```

inference using the the true network confusion matrix for the true network

```
set.seed(123)
results = matrix(nrow=nrow(test_dat),ncol=1)
for (i in 1:nrow(test_dat)){
  row = as.matrix(test_dat[i,])
  prob  = cpquery(true_fit,
          event = S=="yes",
```

```
          evidence = (A==test_dat[i,1]) & (T==test_dat[i,3]) & (L==test_dat[i,4]) & (B==test_dat[i,5]) &
          )

  results[i] = prob
  }



conf_matrix = table(results>=.5,test_dat$S)
accuracy = (conf_matrix[1,1]  + conf_matrix[2,2]) / sum(conf_matrix)
print(conf_matrix)
```

```
##
##          no yes
##   FALSE 361 118
##   TRUE  139 382
```

```
print(paste("accuracy:",accuracy))
```

```
## [1] "accuracy: 0.743"
```

the learned and non-true network shows slighty results.

Question 3

Identifying blanket

```
my_arcs = mod_80$arcs
my_arcs = as.matrix(my_arcs)

parents = my_arcs[which(my_arcs[,2]=="S"),1]

children = my_arcs[which(my_arcs[,1]=="S"),2]

parents_of_children = my_arcs[which(my_arcs[,2] %in%children),1]

blanket = unique(c(parents, children, parents_of_children))


blanket = blanket[(blanket!="S")]

print(blanket)
```

```
## [1] "B" "L"
```

```
mb(mod_80,"S")
```

```
## [1] "L" "B"
```

```
# P.S. noticed the instruction too late, at least they are the same :)
```

Inference and accuracy using only thee markov blanket.

```
set.seed(123)
results = matrix(nrow=nrow(test_dat),ncol=1)
  for (i in 1:nrow(test_dat)){
    row = as.matrix(test_dat[i,])
    prob  = cpquery(true_fit,
            event = S=="yes",
            evidence = (L==test_dat[i,4]) & (B==test_dat[i,5])

            )

    results[i] = prob
  }


conf_matrix = table(results>=.5,test_dat$S)
accuracy = (conf_matrix[1,1]  + conf_matrix[2,2]) / sum(conf_matrix)
print(conf_matrix)
```

```
##
##          no yes
##   FALSE 359 116
##   TRUE  141 384
```

```
print(paste("accuracy:",accuracy))
```

```
## [1] "accuracy: 0.743"
```

no significant loss of performance.

Qeustion 4

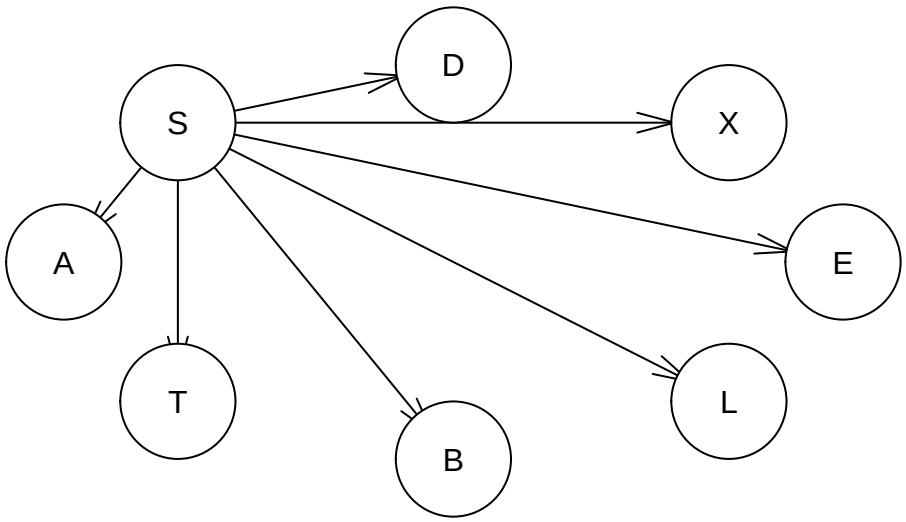
Creating and fitting the Naive Bayes classifier

```
set.seed(123)
target_var = "S"
others = c("A","T","B","L","E","X","D")

nb_arcs = matrix(nrow=length(others),ncol=2)
colnames(nb_arcs) = c("from", "to")

nb_arcs[,1] = target_var
nb_arcs[,2] = others
nb_arcs
```

```
##      from to
```

```
## [1,] "S"  "A"
## [2,] "S"  "T"
## [3,] "S"  "B"
## [4,] "S"  "L"
## [5,] "S"  "E"
## [6,] "S"  "X"
## [7,] "S"  "D"
```

```
my_nb = empty.graph(c(target_var,others))
my_nb$arcs = nb_arcs

#plot(true_net)
plot(my_nb)
```




```
bn_fitted = bn.fit(my_nb,train_dat)
```

Inference with the NB-classifier

```
results = matrix(nrow=nrow(test_dat),ncol=1)
  for (i in 1:nrow(test_dat)){
    row = as.matrix(test_dat[i,])
    prob  = cpquery(bn_fitted,
            event = S=="yes",
            evidence = (A==test_dat[i,1]) & (T==test_dat[i,3]) & (L==test_dat[i,4]) & (B==test_dat[i,5])
            )
```

```
    results[i] = prob
  }


conf_matrix = table(results>=.5,test_dat$S)
accuracy = (conf_matrix[1,1]  + conf_matrix[2,2]) / sum(conf_matrix)
print(conf_matrix)
```

```
##
##          no yes
##   FALSE 210 240
##   TRUE  290 260
```

```
print(paste("accuracy:",accuracy))
```

```
## [1] "accuracy: 0.47"
```

the result is much worse

Question 5

The results are all different due to the arcs and probabilities learned. Usin only the markov blanket during inference made only insignificant difference to the the full network from question 2.

The so,called true network was not materially better than the one trained on 80% of the data.