# Lab1

aswma317

2023-09-06

Question 1: Show that multiple runs of the hill-climbing algorithm can return non-equivalent Bayesian network (BN) structures.

Answer: 2 BNs are said to be equivalent if,

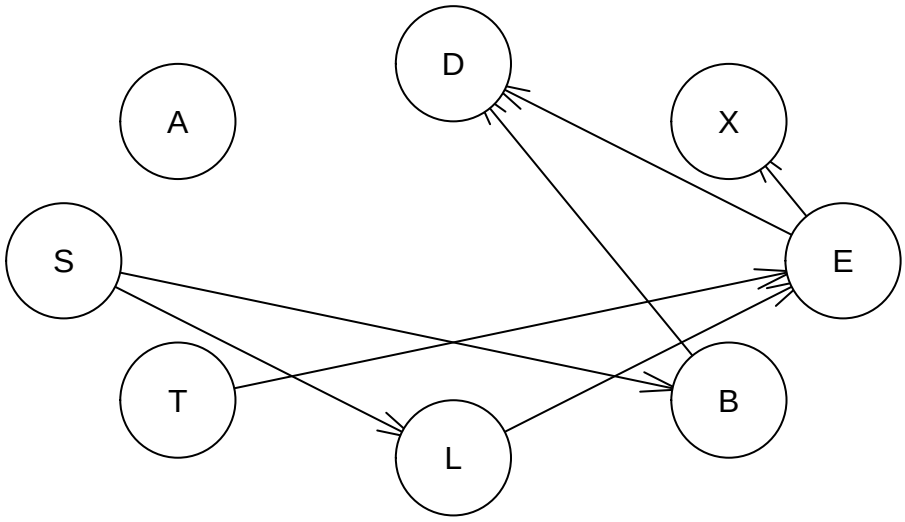
i) they have the same adjancancies(skeleton)

ii) they have the same unshielded collider

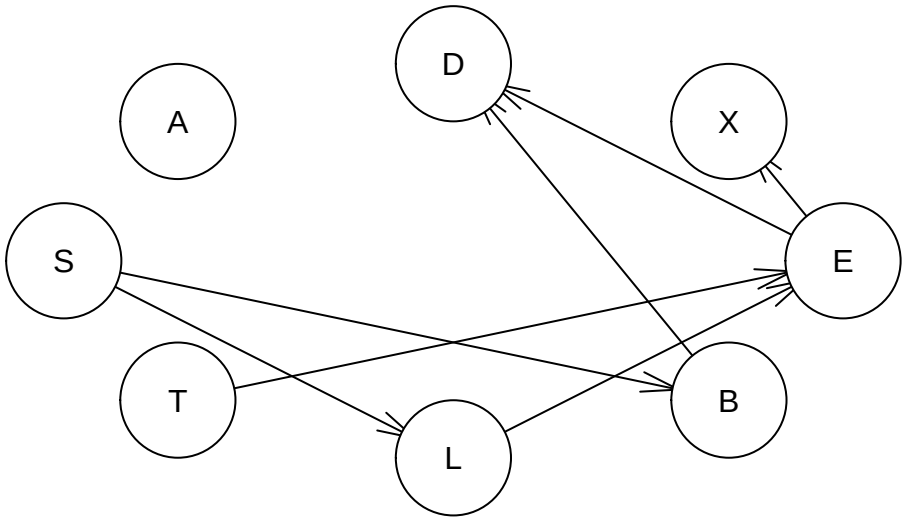
HC starts with an empty DAG and iteratively improves the Bayesian Score by adding/removing/reversing the edges in the DAG. But it may not converge to the optimal solution(true graph) as it may get stuck in the local optimum. Hence, it is not asymptotically correct under faithfulness; i.e. regardless of the number of samples we have, it does not converge to the optimum solution.

This is visible from the below experiment:

```
#library(bnlearn) To create structure from BN
#library(gRain) For probabilistic inference of Graph
graph1 <- bnlearn::hc(asia)
plot(graph1)
```



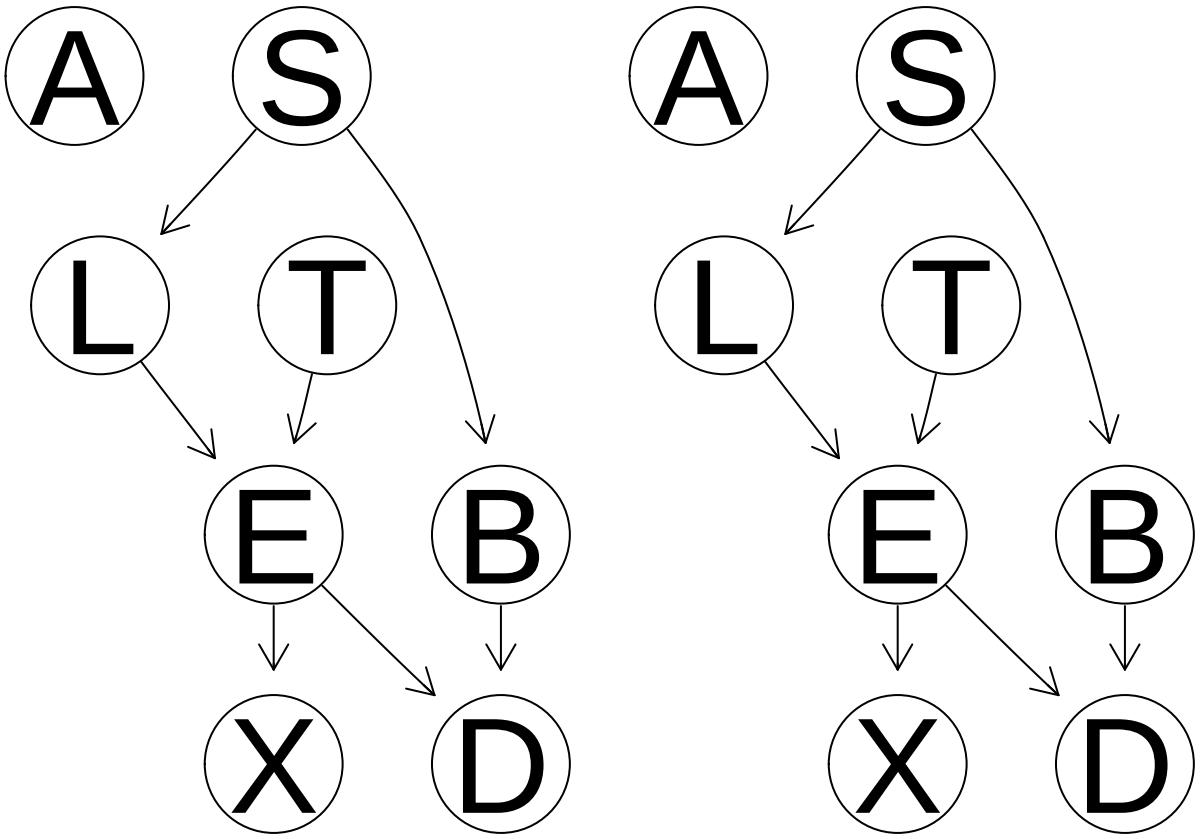

```
#Restart: when it hits the local optimum, it restarts again to explore new areas in the graph.
graph2 <- bnlearn::hc(asia, restart = 5)
plot(graph2)
```




```
print(paste("Graph 1 and Graph 2 equivalency:", all.equal(graph1, graph2)))
```

```
## [1] "Graph 1 and Graph 2 equivalency: TRUE"
```

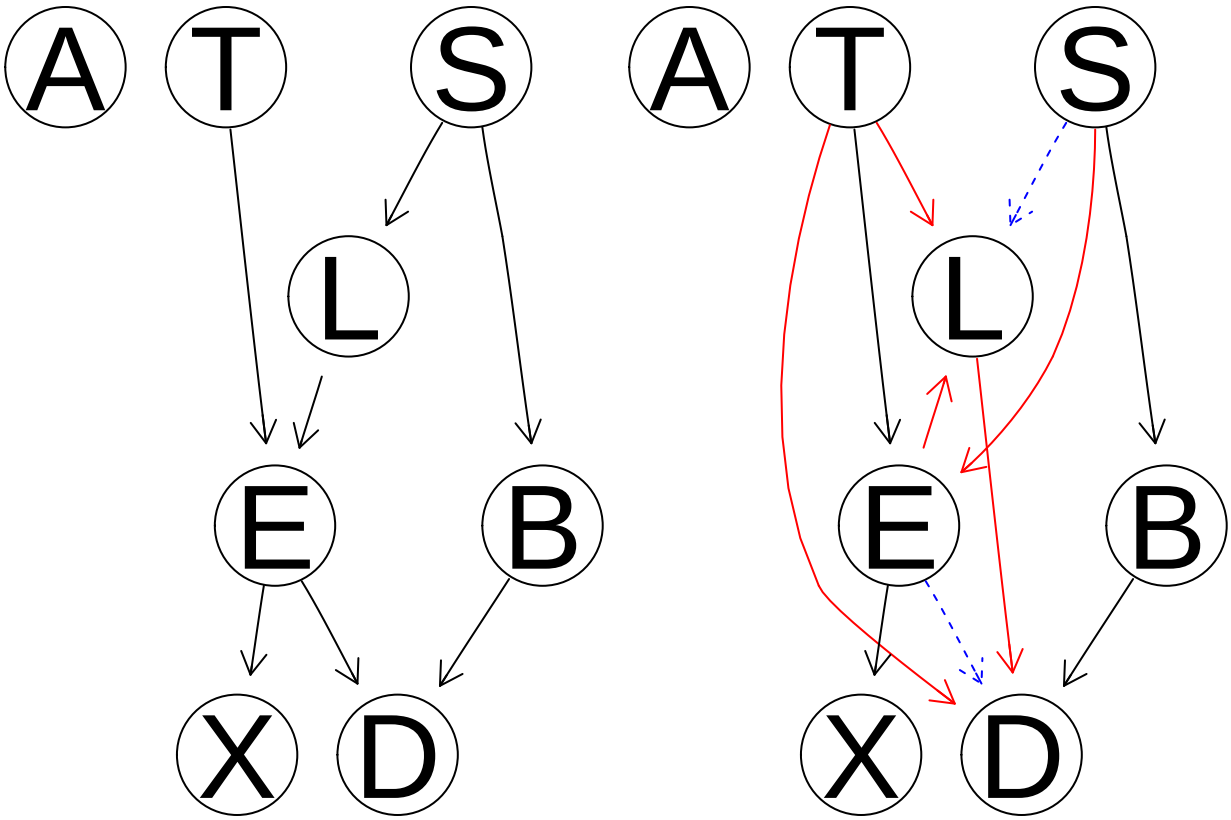
```
par(mfrow = c(1, 2))
graphviz.compare(graph1, graph2)
```



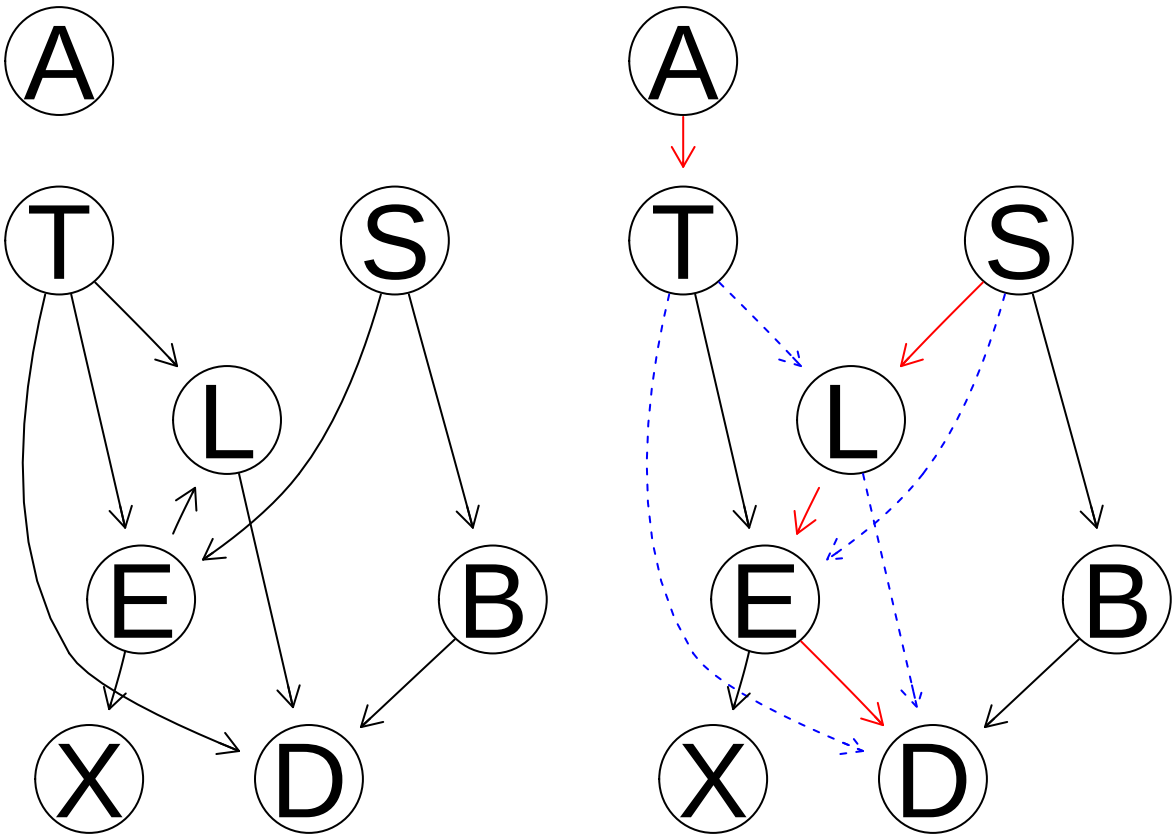

```
#Change the initial structure
custom_initial_structure <- random.graph(names(asia), num = 1, method = "ordered")
graph3 <- bnlearn::hc(asia, start = custom_initial_structure)
print(paste("Graph 2 and Graph 3 equivalency:", all.equal(graph3, graph2)))
```

```
## [1] "Graph 2 and Graph 3 equivalency: Different number of directed/undirected arcs"
```

```
par(mfrow = c(1, 2))
graphviz.compare(graph1, graph3)
```




```
#Change the scoring
graph4 <- bnlearn::hc(asia, score = "bde", iss = 2)
graphviz.compare(graph3, graph4)
```

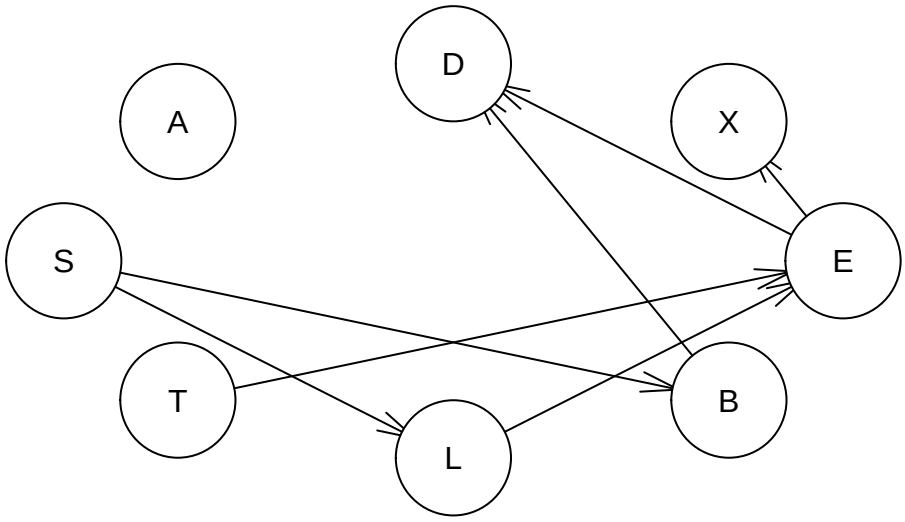



In the above experiment, we changed the parameters like the restarts and the starting point of the graph. But, HC doesn't guarantee equivalent BN.

Question 2: Compute the posterior probability distribution of S for each case and classify it in the most likely class.

Answer:

```
#Split the data
set.seed(123456789)
train_idx <- sample(1:nrow(asia), size = nrow(asia) * 0.8)
data_train <- asia[train_idx,]
data_test <- asia[-train_idx,]

#Learn the structure
learned_graph <- bnlearn::hc(data_train)
plot(learned_graph)
```



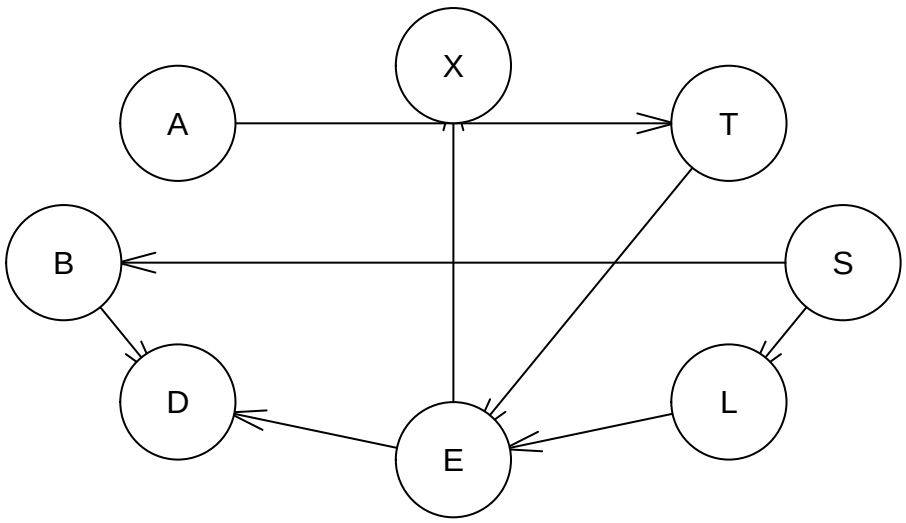

```
#Learn the parameters/conditional probability distributions
bn_fit <- bn.fit(learned_graph, data = data_train)
#method = "bayes" for imputing non NA values
#custom.fit can be used if data is not available and the conditional prob is known

#Perform probabilistic inference like computing posterior probabilities,
#conditional probabilities, classifications ...
#Convert the bn.fit object to a grain
#bn_grain <- as.grain(bn_fit)
junction = compile(as.grain(bn_fit))

pred_exact_inf <- character(length = nrow(data_test))
for (i in 1:nrow(data_test)) {
  # Create an evidence object with the observed variables from the test data
  evidence <- setEvidence(junction,
                          nodes = c("A", "T", "L", "B", "E", "X", "D"),
                          states = t(data_test[i ,-2]))
  #When there is no evidence
  #setEvidence(junction,nodes=c(""),states=c(""))

  # Query the posterior probability of S(S = yes and S = no)
  posterior <- querygrain(evidence,
                          nodes = "S",
                          type = "marginal")

  # Predict the correct class
```

```
  pred_exact_inf[i] <- ifelse(posterior[[1]][1] > posterior[[1]][2],
                              "no", "yes")
  #Can be sampled as well
  #sample(c("no","yes"),size=1,prob=posterior$S)
}

#Confusion matrix
true_labels <- as.vector(data_test$S)
confusion_matrix <- table(Actual = true_labels, Predicted = pred_exact_inf)
print(paste("Accuracy of posterior prediction using exact inference",
            sum(diag(confusion_matrix))/sum(confusion_matrix)))
```

```
## [1] "Accuracy of posterior prediction using exact inference 0.719"
```

Comparing with the true BN

```
#Compare with the true DAG
true_dag = model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")
plot(true_dag)
```



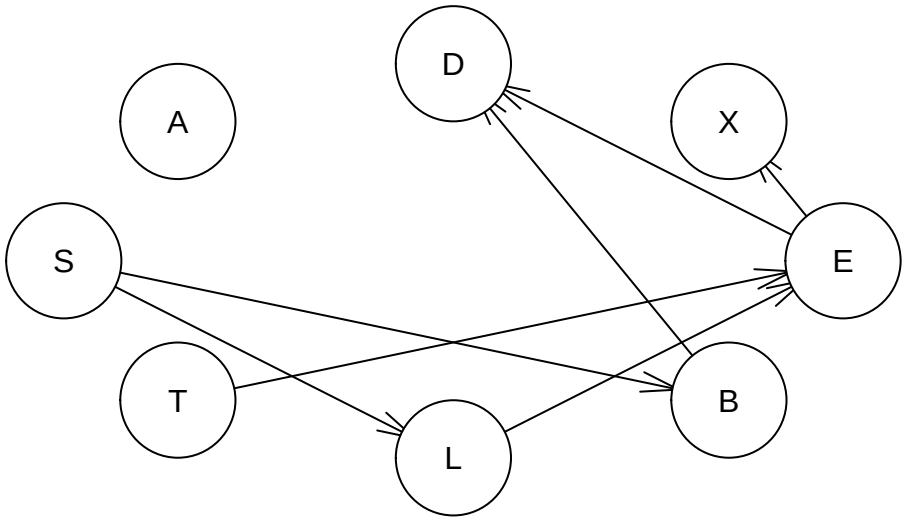

```
#Learn the parameters/conditional probability distributions for true dag
bn_fit_true <- bn.fit(true_dag, data = data_train)
junction = compile(as.grain(bn_fit_true))
pred_true_dag <- character(length = nrow(data_test))
```

```r
#We can also sample new values from the conditional probabilities
#a <- sample(1:2, 1, prob = bn_fit_true$A$prob)
for (i in 1:nrow(data_test)) {
  # Create an evidence object with the observed variables from the test data
  evidence <- setEvidence(junction,
                          nodes = c("A", "T", "L", "B", "E", "X", "D"),
                          states = t(data_test[i ,-2]))

  # Query the posterior probability of S(S = yes and S = no)
  posterior <- querygrain(evidence,
                          nodes = "S",
                          type = "marginal")

  # Predict the correct class
  pred_true_dag[i] <- ifelse(posterior[[1]][1] > posterior[[1]][2],
                             "no", "yes")
}

#Confusion matrix
true_labels <- as.vector(data_test$S)
confusion_matrix_true <- table(Actual = true_labels, Predicted = pred_true_dag)
print(paste("Accuracy of posterior prediction of true DAG",
            sum(diag(confusion_matrix_true))/sum(confusion_matrix_true)))
```

```
## [1] "Accuracy of posterior prediction of true DAG 0.719"
```

Estimating the posterior probability using approximate inference(cpquery):

```r
#Learn the structure
learned_graph <- bnlearn::hc(data_train)
plot(learned_graph)
```

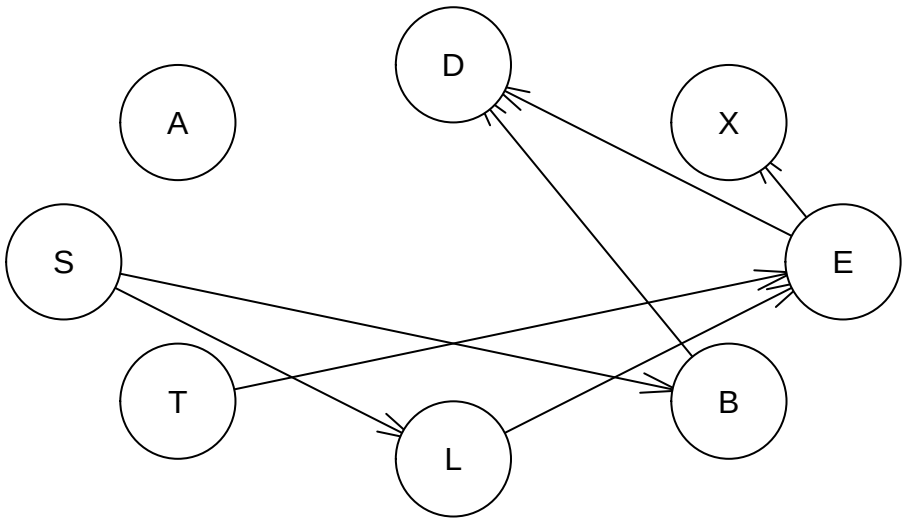



```r
#Learn the parameters/conditional probability distributions
bn_fit <- bn.fit(learned_graph, data = data_train)

#Perform probabilistic inference like computing posterior probabilities,
#conditional probabilities, classifications ...
pred_approx_inf <- character(length = nrow(data_test))
for (i in 1:nrow(data_test)) {
  # Create an evidence object with the observed variables from the test data
  evidence <- data_test[i ,-2]

  # Query the posterior probability of S(S = yes and S = no)
  posterior <- cpquery(bn_fit,
                       event = (S == "yes"),
                       evidence = as.list(evidence),
                       method = "lw")

  # Predict the correct class
  pred_approx_inf[i] <- ifelse(posterior >= 0.5, "yes", "no")
}

#Confusion matrix
true_labels <- as.vector(data_test$S)
confusion_matrix <- table(Actual = true_labels, Predicted = pred_approx_inf)
print(paste("Accuracy of posterior prediction using approx inference",
            sum(diag(confusion_matrix))/sum(confusion_matrix)))
```

```
## [1] "Accuracy of posterior prediction using approx inference 0.719"
```

Estimating the posterior probability using approximate inference(cpdist):

```
#Learn the structure
learned_graph <- bnlearn::hc(data_train)
plot(learned_graph)
```




```
#Learn the parameters/conditional probability distributions
bn_fit <- bn.fit(learned_graph, data = data_train)

#Perform probabilistic inference like computing posterior probabilities,
#conditional probabilities, classifications ...
pred_approx_inf <- character(length = nrow(data_test))
for (i in 1:nrow(data_test)) {
  # Create an evidence object with the observed variables from the test data
  evidence <- data_test[i ,-2]

   # Query the posterior probability of S(S = yes and S = no)
  posterior_sim <- cpdist(bn_fit,
                      nodes = "S",
                       evidence = as.list(evidence),
                       method = "lw")
  #If either event or evidence is set to TRUE an unconditional probability query
  #is performed with respect to that argument.
```

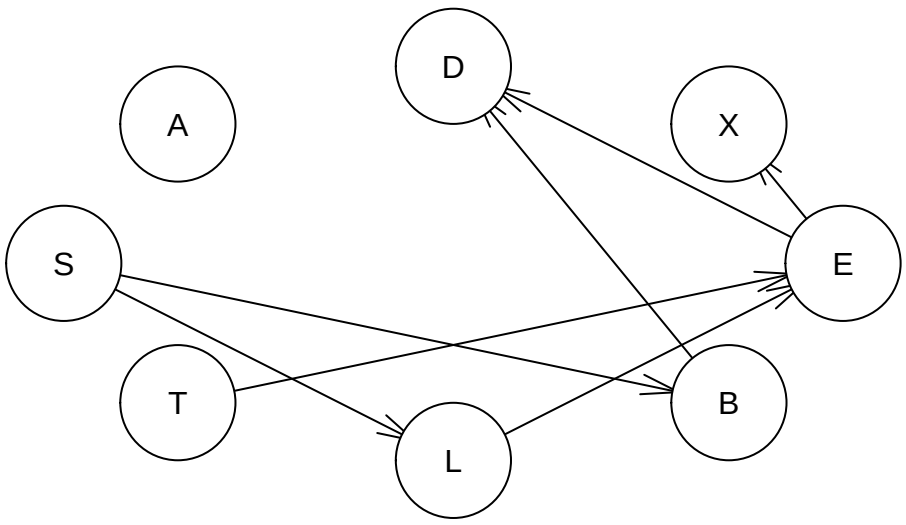
```
  # Predict the correct class
  pred_approx_inf[i] <- ifelse(mean(posterior_sim == 'yes') >= 0.5, "yes", "no")
}

#Confusion matrix
true_labels <- as.vector(data_test$S)
confusion_matrix <- table(Actual = true_labels, Predicted = pred_approx_inf)
print(paste("Accuracy of posterior prediction using approx inference",
            sum(diag(confusion_matrix))/sum(confusion_matrix)))
```

```
## [1] "Accuracy of posterior prediction using approx inference 0.487"
```

Question 3: Classify S given observations only for the so-called Markov blanket of S

Answer:

```
#Learn the structure using training
learned_graph <- bnlearn::hc(data_train)
plot(learned_graph)
```




```
# Identify the Markov blanket of variable S
mb_S <- mb(learned_graph, node = "S")
#Question: No 'D' in MB
```

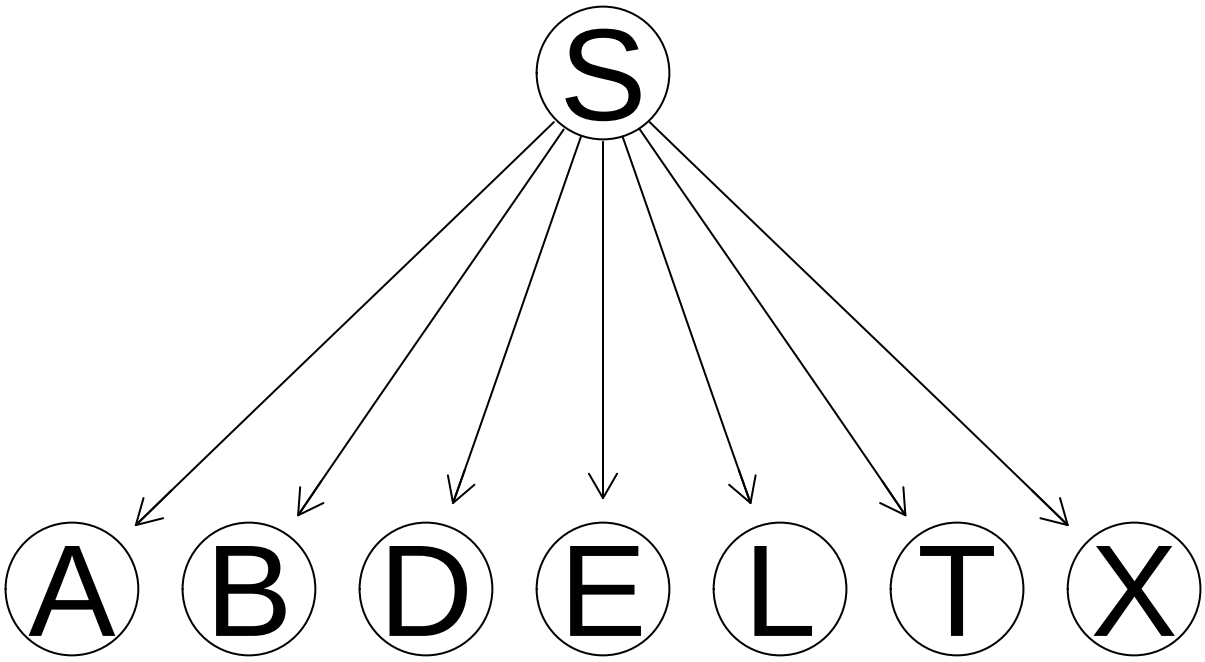
```r
#Learn the parameters/conditional probability distributions
bn_fit <- bn.fit(learned_graph, data = data_train)

#Perform probabilistic inference like computing posterior probabilities,
#conditional probabilities, classifications ...
junction = compile(as.grain(bn_fit))

pred_mb_s <- character(length = nrow(data_test))
for (i in 1:nrow(data_test)) {
  # Create an evidence object with the observed variables from the test data
  evidence <- setEvidence(junction,
                          nodes = names(data_test[i, mb_S]),
                          states = t(data_test[i ,mb_S]))

  # Query the posterior probability of S(S = yes and S = no)
  posterior <- querygrain(evidence,
                          nodes = "S",
                          type = "marginal")

  # Predict the correct class
  pred_mb_s[i] <- ifelse(posterior[[1]][1] > posterior[[1]][2], "no", "yes")
}

#Confusion matrix
true_labels <- as.vector(data_test$S)
confusion_matrix <- table(Actual = true_labels, Predicted = pred_mb_s)
print(paste("Accuracy of posterior prediction conditioned on MB(S):",
            sum(diag(confusion_matrix))/sum(confusion_matrix)))
```

```
## [1] "Accuracy of posterior prediction conditioned on MB(S): 0.719"
```

Question 4: Model the naive Bayes classifier as a BN

Answer:

```r
#S is the parent in Naive Bayes, all observed data is independent when conditioned
#on the class label S
naive_bayes_graph <- model2network("[S][A|S][T|S][L|S][B|S][D|S][E|S][X|S]")
graphviz.plot(naive_bayes_graph)
```




```r
#Split the data
set.seed(123456789)
train_idx <- sample(1:nrow(asia), size = nrow(asia) * 0.8)
data_train <- asia[train_idx,]
data_test <- asia[-train_idx,]

#Learn the parameters/conditional probability distributions
bn_fit <- bn.fit(naive_bayes_graph, data = data_train)

#Perform probabilistic inference like computing posterior probabilities,
#conditional probabilities, classifications ...
#Convert the bn.fit object to a grain
#bn_grain <- as.grain(bn_fit)
junction = compile(as.grain(bn_fit))

pred_bayes_inf <- character(length = nrow(data_test))
for (i in 1:nrow(data_test)) {
  # Create an evidence object with the observed variables from the test data
  evidence <- setEvidence(junction,
                          nodes = c("A", "T", "L", "B", "E", "X", "D"),
                          states = t(data_test[i ,-2]))

  # Query the posterior probability of S(S = yes and S = no)
  posterior <- querygrain(evidence,
                          nodes = "S",
                          type = "marginal")
```

```
  # Predict the correct class
  pred_bayes_inf[i] <- ifelse(posterior[[1]][1] > posterior[[1]][2],
                              "no", "yes")
}

#Confusion matrix
true_labels <- as.vector(data_test$S)
confusion_matrix <- table(Actual = true_labels, Predicted = pred_bayes_inf)
print(paste("Accuracy of posterior prediction:",
            sum(diag(confusion_matrix))/sum(confusion_matrix)))
```

```
## [1] "Accuracy of posterior prediction: 0.673"
```

Question 5: Explain the difference in results between 2 and 4

The posterior probability distribution of S, given the observed variables when we perform exact or approximate inference as per Question 2 is as follows:

$$p(S|B,L) = \frac{p(S,B,L)}{p(B,L)} = \frac{p(S).p(B|S).p(L|S)}{p(B,L)}$$

In the above expression, we consider only children of S, because it is most likely that they affect the posterior probability distribution of S due to conditional independence property.

In the case of Naive Bayes classifier, the posterior probability of S is as follows:

$$p(S|A,T,L,B,E,X,D) = \frac{p(S,A,T,L,B,E,X,D)}{p(A,T,L,B,E,X,D)}$$
$$= \frac{p(S)\cdot p(A|S)\cdot p(T|S)\cdot p(L|S)\cdot p(B|S)\cdot p(E|S)\cdot p(X|S)\cdot p(D|S)}{p(A,T,L,B,E,X,D)}$$

From the expression above it is evident that the posterior probability of S now relies on all the children A,T,L,B,E,X,D, and hence the difference in accuracy and confusion matrix i.e., we report a lower accuracy compared to previous methods.