LAB1_AML

Andreas C Charitos [andch552]

9/18/2019

Contents

Assignment 1

In this part we are going to use grid search in order to find the best parameters for giving the best DAG structure. The search is not going to be exhausive (we are testing small grids) and we are using

- $$grid\ for\ restart\ (from = 1, to = 10, by = 1)$$
- $$grid\ for\ score\ ("loglik", "aic", "bic", "bdla", "bdj", "bde", "bds", "mbde")$$
- $$grid\ for\ max.iter\ (from = 1, to = 10, by = 1)$$
- $$and\ initial\ random.graph$$

The plot above shows the graph that learned with the best parameters returned from the grid search.

Plot of the Brute Force Network

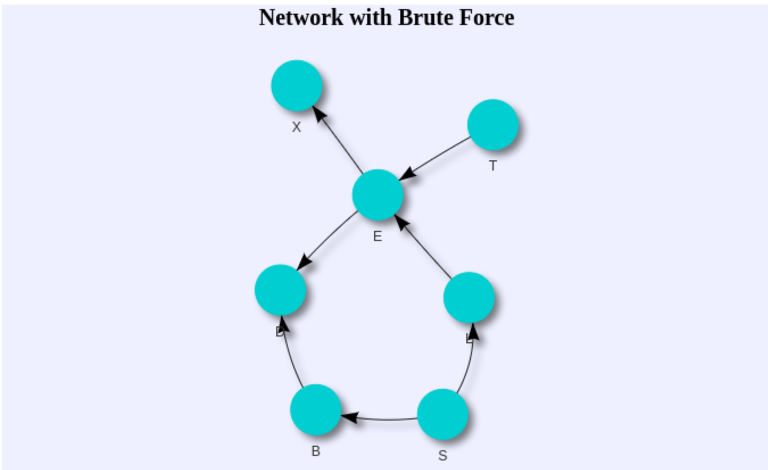


Figure 1: *Network with brute force.*

Best combination output and modelstring

The best combination from the grid search is shown in the table below.

	restart	score	max.iter
775	5	bde	10

Here we can see the string of the model that we get using the brute force method.

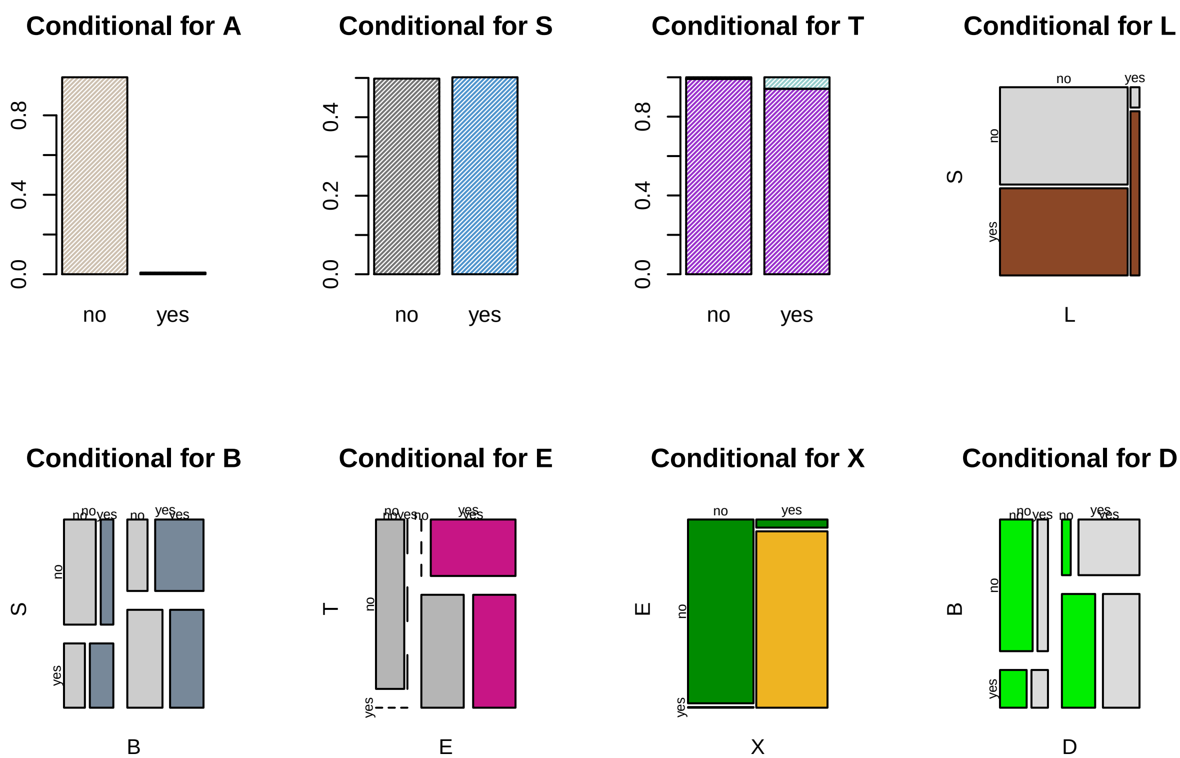
The string model is : [A][S][T][L|S][B|S][E|T:L][X|E][D|B:E]

Assignment 2

In this part using the network structed learned in the previous part we are fitting the 80% of the asia dataset as train data and the remaining 20% as test data in order to make predictions for the node $S(”yes”, ”no”)$ which is the variable Smoking in the asia dataset

Plot of the Conditionals Tables

The plot above gives the conditionals tables for all the nodes in the Graph.

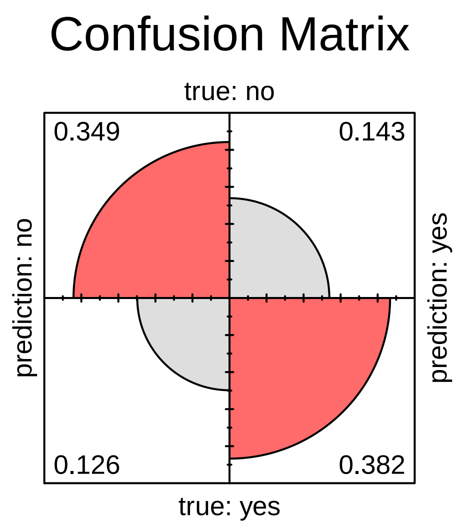


The confusion matrix for the brute force model is :

```
##      prediction
## true     no   yes
##   no  0.349 0.143
##   yes 0.126 0.382
```

Confusion Matrix Plot for Brute Force Network

The plot above shows the confusion matrix for the BF



The accuracy of brute force model is: 0.731

Comparison with True model

We continue the analysis where we compare the model that we have with the true model ("$[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]$")

The plot of network of the true model is given below.

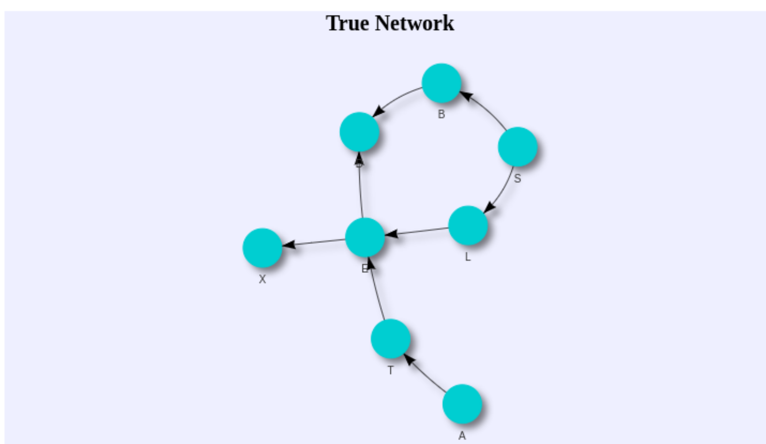


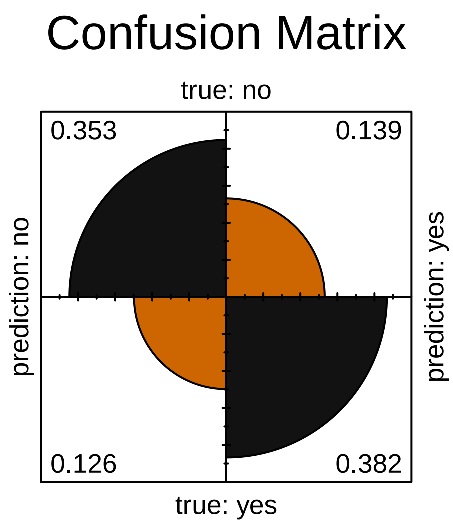
Figure 2: *Network with true model.*

The confusion matrix for the true model is :

```
##      prediction
## true     no   yes
##   no  0.353 0.139
##   yes 0.126 0.382
```

Confusion Matrix Plot for True Network

The confusion matrix plot for the true model is shown below.



The accuracy of true model is: 0.735

Assignment 3

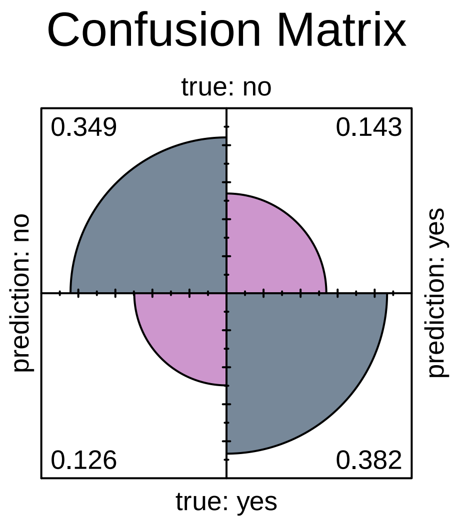
In this part we are going to use the Markov Blankets of the S node to make the predictions. In statistics and machine learning, the Markov blanket for a node in a graphical model contains all the variables that shield the node from the rest of the network. This means that the Markov blanket of a node is the only knowledge needed to predict the behavior of that node and its children. `source Wikepedia [link]`

The confusion matrix for the markov blanket model is :

```
##      prediction
## true     no   yes
##   no  0.349 0.143
##   yes 0.126 0.382
```

Confusion Matrix Plot for Markov Blancket

The confusion matrix plot for the Markov Blanket is shown below.



The accuracy of narkov blanket model is: 0.731

Assignment 4

Finally,we are testing a Naive Bayes model for the node S. In machine learning, naïve Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. `source Wikepedia [link]`

The plot of Naive Bayes network is shown below

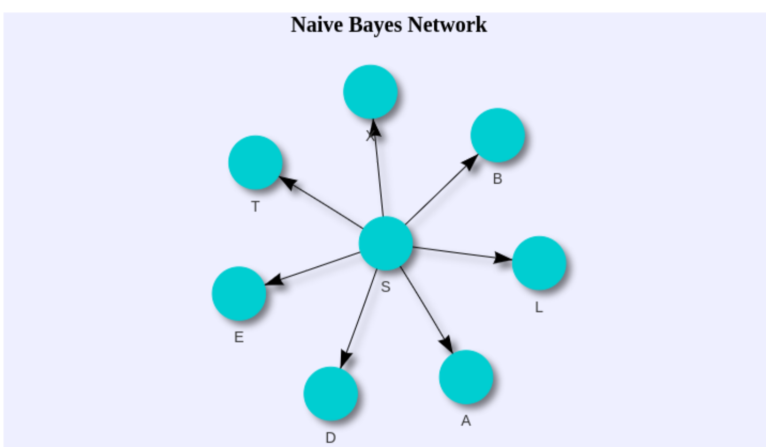


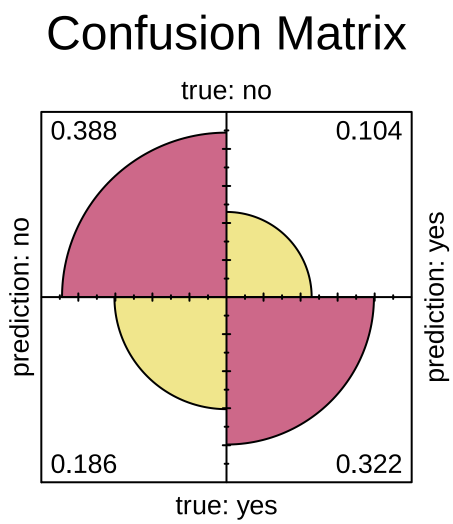
Figure 3: *Network with Naive Bayes.*

The confusion matrix for the naive bayes model is :

```
##      prediction
## true     no   yes
##   no  0.388 0.104
##   yes 0.186 0.322
```

Confusion Matrix Plot for Naive Bayes Network

The plot of the confusion matrix is shown below.



The accuracy of brute force model is: 0.71

Summary Table with accuracies

	Accuracy
BF-model	0.73
True-model	0.74
MB-model	0.73
Naive-model	0.71

Conclusion

In conclusion the Brute Force model although it had different structure that the true has able to achive similar accuracy with the true model. The Markov Blanket model had the same accuracy with the brute force as expected because according to the definition of the Markov Blanket instead of using all the paremeters as with the Brute force inference using the Markov Blankets we use only the parameters that the node is dependent. The Naive Bayes model finally achieved a slightly worse accuracy than the previous models and again this can be explained by the definition of Naive Bayes that assumes cross-independence for all the other nodes.

Appendix

Code used for Lab

```r
# load libraries and data
library(bnlearn)
library(Rgraphviz)
library(visNetwork)
library(igraph)
library(gRain)
data("asia")
set.seed(123456789)
# Assignment 1
# ------------------------------------------------------------
plot.network <- function(structure, ht = "400px", my_title) {
    # https://www.r-bloggers.com/bayesian-network-example-with-the-bnlearn-package/
    nodes.uniq <- unique(c(structure$arcs[, 1], structure$arcs[,
        2]))
    nodes <- data.frame(id = nodes.uniq, label = nodes.uniq,
        color = "darkturquoise", shadow = TRUE)
    edges <- data.frame(from = structure$arcs[, 1], to = structure$arcs[,
        2], arrows = "to", smooth = TRUE, shadow = TRUE,
        color = "black")
    return(visNetwork(nodes, edges, height = ht, width = "100%",
        background = "#eeefff", main = my_title))
}
restartGrid <- seq(1, 10, 1)
scoreGrid <- c("loglik", "aic", "bic", "bdla", "bdj", "bde",
    "bds", "mbde")
max.iterGrid <- seq(1, 10, 1)
rnd <- random.graph(nodes = colnames(asia))
gridMatrix <- expand.grid(restartGrid, scoreGrid, max.iterGrid)
colnames(gridMatrix) <- c("restart", "score", "max.iter")
gridMatrix[, 2] <- as.character(gridMatrix[, 2])
xs <- apply(gridMatrix, 1, function(x) {
    hc(asia, restart = as.numeric(x[1]), score = x[2], max.iter = as.numeric(x[3]),
        star = rnd)
})
xx <- lapply(xs, function(y) {
    bnlearn::score(y, asia)
})
best_index = which.max(unlist(xx))
best_combination <- gridMatrix[best_index, ]
knitr::kable(best_combination)
cat("The string model is :\n")
modelstring(xs[[best_index]])
# Assignment 2
# ------------------------------------------------------------
## 80% of the sample size
smp_size <- floor(0.8 * nrow(asia))
asia_character <- data.frame(lapply(asia, as.character),
    stringsAsFactors = FALSE)  # create a df only for to use in the setEvidence function
indx <- sample(nrow(asia), size = smp_size)
```

```r
asia_test <- asia_character[-indx, ]  # use this only for the states arg in setEvidence
train_data <- asia[indx, ]
test_data <- asia[-indx, ]
dag.fit <- hc(train_data, restart = 5, score = "bde", max.iter = 10)  # learn structure
bn.model <- bn.fit(dag.fit, data = train_data, method = "mle")  # fit the model-learn the parameters
bn.model.grain <- compile(as.grain(bn.model))  # compile as grain object
col.param = length(colnames(asia))/2
par(mfrow = c(2, col.param))
for (i in 1:length(bn.model)) {
    obj <- bn.model[[i]]$prob
    if (names(bn.model)[i] %in% c("A", "S", "T")) {
        barplot(obj, col = sample(colors()), density = 60,
            main = paste("Conditional for", names(bn.model)[i]))
    } else {
        plot(obj, col = sample(colors()), main = paste("Conditional for",
            names(bn.model)[i]))
    }
    ""
}
prediction_func <- function(fit.dag, train_data, test_data,
    method, index, node) {
    fit.model <- bn.fit(fit.dag, data = train_data, method = method)  # fit the model-learn the paramet
    fit.model.grain <- compile(as.grain(fit.model))
    pred_vector <- double(dim(test_data)[1])
    for (i in 1:dim(test_data)[1]) {
        evidence.obj <- setEvidence(fit.model.grain, nodes = colnames(test_data[-index]),
            states = asia_test[i, -index])
        query.obj <- querygrain(evidence.obj, nodes = node,
            type = "marginal")
        pred_vector[i] <- ifelse(query.obj[[1]][1] > query.obj[[1]][2],
            "no", "yes")
    }
    return(pred_vector)
}
pred_bf <- prediction_func(dag.fit, train_data, test_data,
    "mle", 2, "S")
tab_bf <- prop.table(table(test_data[, 2], pred_bf, dnn = c("true",
    "prediction")))
cat("The confusion matrix for the brute force model is :\n")
tab_bf
fourfoldplot(tab_bf, color = c(sample(colours(), 1), sample(colours(),
    1)), conf.level = 0, margin = 1, main = "Confusion Matrix")
accuracy_bf <- sum(pred_bf == test_data[, 2])/dim(test_data)[1]
cat("The accuracy of brute force model is:", accuracy_bf *
    100, "%")
dag.true = model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")
pred_true <- prediction_func(dag.true, train_data, test_data,
    "mle", 2, "S")
tab_true <- prop.table(table(test_data[, 2], pred_true,
    dnn = c("true", "prediction")))
cat("The confusion matrix for the brute force model is :\n")
tab_true
fourfoldplot(tab_true, color = c(sample(colours(), 1), sample(colours(),
```

```r
    1)), conf.level = 0, margin = 1, main = "Confusion Matrix")
accuracy_true <- sum(pred_true == test_data[, 2])/dim(test_data)[1]
cat("The accuracy of true model is:", accuracy_true * 100,
    "%")
# Assignment 3
# ------------------------------------------------------------
mv.blanket <- bnlearn::mb(dag.fit, "S")  # markov blancket for S
indexes <- c()
for (obj in mv.blanket) {
    ind <- which(obj == colnames(test_data))
    indexes <- c(indexes, ind)
}
r <- seq(1:8)
indexes <- r[-indexes]
# ----------------------------------------------------------
pred_mb <- prediction_func(dag.fit, train_data, test_data,
    "mle", indexes, "S")
tab_mb <- prop.table(table(test_data[, 2], pred_mb, dnn = c("true",
    "prediction")))
cat("The confusion matrix for the markov blancket model is :\n")
tab_mb
fourfoldplot(tab_mb, color = c(sample(colours(), 1), sample(colours(),
    1)), conf.level = 0, margin = 1, main = "Confusion Matrix")
accuracy_mb <- sum(pred_mb == test_data[, 2])/dim(test_data)[1]
cat("The accuracy of markov blancket model is:", accuracy_mb *
    100, "%")
# Assignment 4
# ------------------------------------------------------------
string <- ""
for (i in 1:length(colnames(asia))) {
    if (colnames(asia)[i] != "S") {
        str2 <- paste("[", colnames(asia)[i], "|S]", sep = "")
    } else {
        str2 <- "[S]"
    }
    string <- paste(string, str2, sep = "")
}
naive.dag <- model2network(string)
pred_naive <- prediction_func(naive.dag, train_data, test_data,
    "mle", 2, "S")
tab_naive <- prop.table(table(test_data[, 2], pred_naive,
    dnn = c("true", "prediction")))
cat("The confusion matrix for the naive bayes model is :\n")
tab_naive
fourfoldplot(tab_naive, color = c(sample(colours(), 1),
    sample(colours(), 1)), conf.level = 0, margin = 1, main = "Confusion Matrix")
accuracy_naive <- sum(pred_naive == test_data[, 2])/dim(test_data)[1]
cat("The accuracy of the naive bayes model is:", accuracy_naive *
    100, "%")
library(knitr)
accuracy_df <- rbind(accuracy_bf, accuracy_true, accuracy_mb,
    accuracy_naive)
accuracy_df <- as.data.frame(accuracy_df)
```

```r
rownames(accuracy_df) <- c("Accuracy BF", "Accuracy True",
    "Accuracy MB", "Accuracy Naive")
colnames(accuracy_df) <- "Summary Table"
kable(accuracy_df, digits = 2, caption = "Accuracy Table.")
# Code to plot with graphviz
# ----------------------------------------------
arc.set <- arcs(naive.dag)
highlight.opts <- list(nodes = colnames(asia), col = sample(colors(),
    1), fill = sample(colors(), 1), arcs = arc.set, lty = 5,
    lwd = 2)
graphviz.plot(naive.model, highlight = highlight.opts, layout = "neato")
```