

STATISTICAL COMPUTATION

BAYESIAN NETWORKS ASSIGNMENT

NAME - HARSH PATHAK
ROLL NUMBER. - 2016041

About the Dataset -

The following seven attributes are chosen for learning the Bayesian Networks.

1. age - age in years and is taken as a continuous variable
2. sex - discrete variable, with 0 = female and 1 = male
3. cp - chest pain type, discrete variable with 4 types
 - Value 1: typical angina
 - Value 2: atypical angina
 - Value 3: non-anginal pain
 - Value 4: asymptomatic
4. trestbps - resting blood pressure (mm Hg), taken as a continuous variable
5. chol - serum cholesterol in mg/dl, continuous variable
6. fbs - fasting blood sugar > 120 mg/dl (1 = true; 0 = false)
7. restecg resting electrocardiographic results
 - o Value 0: normal
 - o Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
 - o Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria

The final outcome is "num" attribute which is a binary variable.

num: diagnosis of heart disease (angiographic disease status)

-- Value 0: < 50% diameter narrowing

-- Value 1: > 50% diameter narrowing

The dataset has 164 cases of patients with type-0 and has 139 cases of type-1 num attributes. Hence the dataset is slightly imbalanced.

We use a train-test split with 80% in train set and 20% in test set.

Structure Learning -

The first part involves learning the structure of the graph from the training data. We use a type of score based learning algorithm known as the hill climbing algorithm for structure learning.

- All score based structure learning algorithms compute a score for various candidate graph structures, and finally select the graph structure which has the maximum score.
- The most common scoring function is the log likelihood score. However, this can lead to overfitting, since a maximal DAG (networks in which no more edges can be added) will always have the highest likelihood score.
- Other scoring functions such as AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) also exist. These model selection criteria can penalise more complex models and hence prevent overfitting.
- Akaike Information criterion for a model M is described as follows -

$$\text{maximise } \log p(X | \theta, M) - |\theta|$$

where $\log p(X | \theta, M)$ is log likelihood and $|\theta|$ is the number of parameters

- Bayesian Information criterion for a model M is now defined as -

$$\text{maximise } BIC(x_n; M) = \log p(x_n | \theta_o, M) - (|\theta|/2)\log(n)$$

where $\log p(x_n | \theta_o, M)$ is the maximum log likelihood
 n is the sample size of dataset
 $|\theta|$ is the number of parameters

- The main idea is as follows. Let's say we use the BIC criterion. We now perform an exhaustive search over all possible graph structures and pick a graph which has the maximum BIC criterion.
- Since the number of graphs is exponential, this becomes infeasible as the number of nodes increases. Hill Climbing is an approximation algorithm that reduces the search space of DAG's by making single edge additions, removals and reversals to current graph structure (initial structure is graph with no edges) at each step.

Proposed Method -

We use the hill climbing approach and compare three different model criteria, log likelihood, AIC and BIC.

Once the graph structure is learnt, we use MLE to estimate the parameters of the graph structure.

Dealing with continuous and missing values -

Since our data is hybrid, we use a variant of Gaussian Bayesian Networks, which supports both discrete and continuous nodes.

1. If a node x is discrete, it's parents need to be discrete as well.
2. If a node x is continuous and has parent set U , then $P(x|U)$ is modeled using linear regression

$$P(x|U) = N(\beta_0 + \beta_1 u_1 + \beta_2 u_2 + \dots + \beta_k u_k, \sigma^2)$$

For the MLE approach, there exists a closed form solution for finding the parameters of the bayesian network.

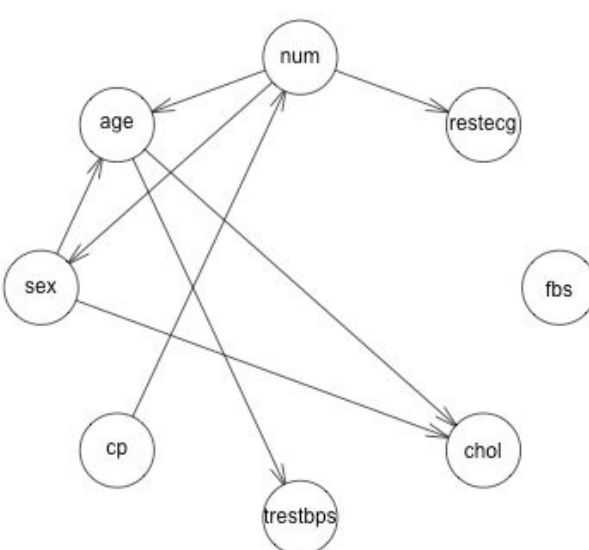
Our seven attributes chosen do not have missing values. Infact, only attributes "ca" and "thal" contain missing values, and there are only 6 missing values in the whole data.

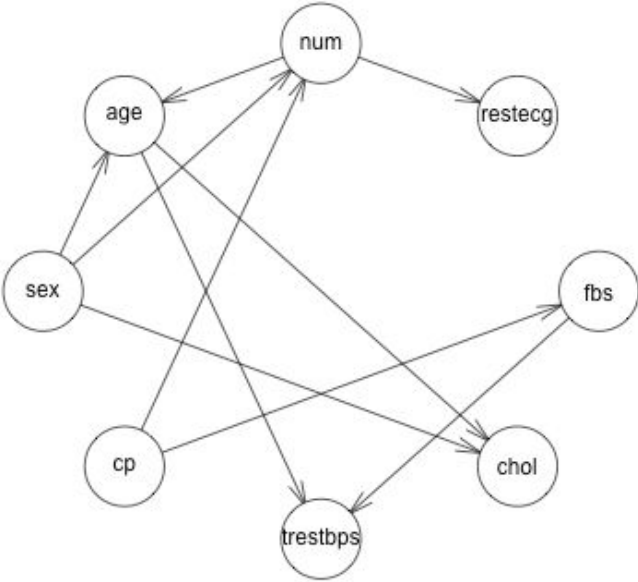
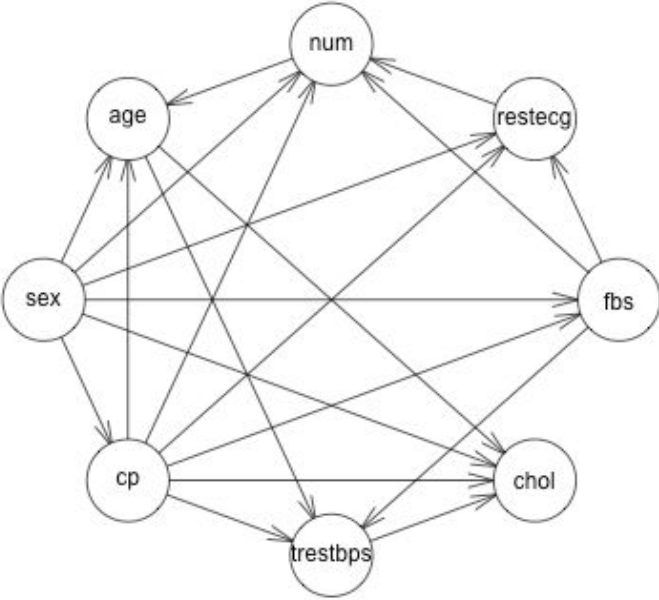
In case missing values are present in the dataset, we can use expectation maximization approach for dealing with them. But we don't require EM algorithm here.

Loopholes with approach -

Since hill climbing is an approximation algorithm, it can output locally optimum solutions that are not globally optimum. However, it has been seen that hill climbing works well in practice for small and medium sized graphs.

Results -

<p>BIC Criterion (hill climbing)</p>  <pre> graph TD age((age)) --> num((num)) age((age)) --> sex((sex)) age((age)) --> cp((cp)) age((age)) --> chol((chol)) num((num)) --> restecg((restecg)) num((num)) --> chol((chol)) sex((sex)) --> num((num)) sex((sex)) --> chol((chol)) cp((cp)) --> chol((chol)) trestbps((trestbps)) --> chol((chol)) fbs((fbs)) </pre>	<p>Training set accuracy 76.44628%</p> <p>Test set accuracy 73.77049%</p> <p>Markov Blanket of num is {cp, sex, age, restecg}</p>
---	---

 <p>AIC criterion (Hill climbing)</p>	<p>Training set accuracy 76.44628%</p> <p>Test set accuracy 73.77049%</p> <p>Markov blanket of num is {age, sex, cp, restecg}</p>
<p>Log likelihood criterion (Hill climbing)</p> 	<p>Training set accuracy 79.75207%</p> <p>Test set accuracy 63.93443%</p> <p>It can be seen that the network structure learnt from log likelihood criterion overfits the data and doesn't generalise well to test data</p> <p>This leads to overfitting, since a maximal DAG (networks in which no more edges can be added) will always have the highest likelihood score.</p> <p>Markov blanket of num is {sex, cp, fbs, restecg, age}</p>

Overview of code

1. Make sure bnlearn is installed.
2. Run the code **Rscript code.R**
3. All the structures will get stored in the same directory
4. The code contains comments for easy understanding

```

library(bnlearn)
set.seed(0)
heart_data = read.csv(file = 'processed.cleveland.data')

use_attributes = 7
discrete_attributes_index = c(2,3,6,7,9,11,12,13)
# use the first 7 attributes,
# age, sex, cp, trestbps, chol, fbs, restecg
# outcome variable is num
labels = heart_data[,14]
#set labels which are 1,2,3,4 to 1, we will distinguish between absence(0)
#and presence(1,2,3,4)
index = labels != 0
labels[index] = 1
labels = factor(labels)

heart_data = heart_data[,1:use_attributes]
heart_data$num = labels
#heart_data is a dataframe containing the following attributes
# age sex cp trestbps chol fbs restecg num

#since columns named sex, cp, fbs and restecg contain discrete values,
#we need to call factor() on those columns
for(id in discrete_attributes_index){
  if(id > use_attributes){
    break
  }
  heart_data[,id] = factor(heart_data[,id])
}
#create train and test sets
#80/20 train test split
train_index <- sample(1:nrow(heart_data), 0.8 * nrow(heart_data))
test_index <- setdiff(1:nrow(heart_data), train_index)

x_train = heart_data[train_index, -use_attributes-2]
y_train = heart_data[train_index, "num"]
x_test = heart_data[test_index, -use_attributes-2]
y_test = heart_data[test_index, "num"]
#NOTE - the first 7 columns do not have missing values
#specify model selection criterions, loglik-cg will
#output a maximal dag, since it has highest likelihood, but
#can overfit
#aic-cg and bic-cg will introduce penalisations(more in report)
model_selection = c("bic-cg", "aic-cg", "loglik-cg")

```

```

for(selection in model_selection){
  filename = paste(selection, "png", sep = ".")
  #use hill climbing for structure learning
  net = hc(x_train, score = selection)

  #plot the network
  png(filename = filename)
  plot(net)
  dev.off()

  #fit the network using MLE
  fitted = bn.fit(net, x_train)

  #predict on train and test to compute accuracy
  train_predict = predict(fitted, "num", x_train)
  test_predict = predict(fitted, "num", x_test)
  acc_train = sum(train_predict == y_train, na.rm = T) / length(train_predict) * 100
  acc_test = sum(test_predict == y_test, na.rm = T) / length(test_predict) * 100
  print(paste("train and test accuracy for", selection, sep = " "))
  print(acc_train)
  print(acc_test)
}

```