Bayesian Networks and Causal Inference

Assignment 2: Learning Bayesian Networks from Data

Guillem Soler Sanz Diego Bartoli Geijo Marta Españó López

27th of January 2023

---

1 Introduction

The aim of this assignment is to compare *Bayesian networks* learnt with two different *structure learning* algorithms between each other and with a hand-constructed network. The task of structure learning for Bayesian networks refers to learning the structure of the directed acyclic graph (DAG) directly from data. There are two major approaches for structure learning [1]:

- *Constraint based algorithms*: these algorithms learn the network structure by analyzing a set of conditional independencies between the variables, then deriving the probabilistic relations entailed by the Markov property of Bayesian networks and then constructing a graph which satisfies the corresponding d-separation statements.
- *Score based algorithms*: these algorithms assign a score to each candidate Bayesian network and try to maximize it with some heuristic search procedure.

In order to explore both approaches we are going to use the *PC* algorithm, constraint based, and the *Hill-Climbing* algorithm, score based. The algorithms' procedures are explained in subsections 1.2 and 1.3. These algorithms don't impose specific restrictions so they can be used with our data, described in subsection 1.1.

1.1 Data

The data used is a subset of the 'Life Expectancy (WHO)' [1] dataset, avalaible on Kaggle. This dataset presents health and economic data for 183 countries from 2000 to 2016. The original dataset consists of 3111 rows and 32 variables. Based on personal evaluation of which variables would be more useful and interesting, we choose 7 variables to start working with. After removing missing values we have 2980 rows. The variables used are reported in the table below with the type and the value range.

Name	Type	% Range
Life expectancy	continous	36.30 - 84.17
Adult Mortality	continous	49.2 - 696.6
Infant Mortality	continous	0.00 - 0.16
Alcohol	continous	0.00 - 20.18
BMI	continous	19.80 - 32.20
Polio	continous	8.00 - 99.00
Domestic GGHE	continous	0.06 - 12.06

[1] [https://www.kaggle.com/datasets/mmattson/who-national-life-expectancy], the dataset is the same used in the first assignment

1.2 The PC algorithm

PC algorithm for structure learning of Bayesian Networks is a constraint based algorithm. From a list of conditional independence statements that hold in a probability distribution P, it produces a CPDAG G such that P is faithful to all DAGs represented by G. Based on the *Lecture Notes* of the course we provide a description of the algorithm.

PC algorithm

---

1. Start with a fully connected graph.
2. For each pair of variables X and Y, search for a set $Z_{X,Y}$ such that the independence $X_{|}|_{Y}|Z_{X,Y}$ is either in the input list or follows from those in the input list. If any such set exists, remove the edge between X and Y.
3. For each pair of variables X and Y that are not linked but have a common neighbour W, check whether $W \in Z_{X,Y}$. If not, then add arrowheads pointing to W.
4. Orient the resulting graph pattern into a CPDAG.

1.3 The Hill-Climbing algorithm

Hill climbing for structure learning of Bayesian Networks is a greedy score based algorithm. A hill climbing greedy search explores the space of the directed acyclic graphs by single-arc addition, removal and reversals; random restarts can be set to avoid local optima. Based on [2] we provide a description of the algorithm procedure:

Hill-Climbing algorithm

---

1. Given a dataset D, a set of nodes V and a set of edges E, initialise:
   (a) $E \leftarrow \emptyset$
   (b) $T \leftarrow$ Conditional Probabilities(E, D)
   (c) $B \leftarrow (V, E, T)$ Bayesian Network
   (d) Score, maximum score $\leftarrow -\infty$
2. Do:
   (a) Maximum score $\leftarrow$ score
   (b) For each set (X, Y) of variables
   and each $E' \in \{E \cup \{X \rightarrow Y\}, E - \{X \rightarrow Y\}, E - \{X \rightarrow Y\} \cup \{Y \rightarrow X\}\}$ such that
   given $T' \leftarrow$ Conditional Probabilities(E', D) then $B' \leftarrow (V, E', T')$ is a B.N., do:
      i. newscore $\leftarrow$ scoringFunction(B', D)
      ii. if newscore $>$ score then $B \leftarrow B'$ and score $\leftarrow$ newscore
3. If score $>$ maximum score return to 2 otherwise return B

The resulting network depends on the scoring function used. Two scoring functions are tested, the *Akaike Information Criterion* and the *Bayesian Information Criterion*, defined as follows:

$$AIC(G|D) = LL(G|D) - |G| \tag{1}$$

$$BIC(G|D) = LL(G|D) - \frac{1}{2} log(|D|)|G| \tag{2}$$

Where G is the graph structure, D is the dataset, and $LL(G|D)$ is the log-likelihood of the graph structure G under the data D. $|D|$ is the number of data samples, and $|G|$ is the number of parameters in the graph. By considering just the first term the optimal graph would be a fully connected graph, probably overfitting the data. The second term serves as a regularization term, favoring simpler models.

2 Methods

2.1 PC algorithm

For the PC algorithm the function *pc* of the *pcalg* package was used. Following the indications provided by the reference manual [2] the parameter *indepTest* was set to to *gaussCItest*, testing conditonal independence between gaussian variables, and the necessary informations for the tests were provided with the parameter *suffStat*. The parameter *alpha* was set first to 0.01 and then to 0.05 to observe the networks obtained with 2 different significance levels for the independency tests. The code used is reported in Appendix B.

2.2 Hill-Climbing algorithm

For the hill climbing algorithm the function *hc*[3] from the *R* package *blearn* [4] is used. The function uses an empty graph as default starting point. The function parameters used are *restart*, number of restarts to avoid local optima, and *score*, the scoring function to use. The first one is arbitrary set to 2, the second one is set to *aic-g* and *bic-g* to implement respectively the Akaike Information Criterion and the Bayesian Information Criterion as scoring functions. The code used is reported in Appendix C.

2.3 Comparison metrics

Structure learning with latent variables is quite complex. Taking into consideration that the algorithm won't actually discover them and add them to the graph, we have decided not to pay as much attention to them. Instead, focus on assessing how the two implemented algorithms perform different structure learning in our data, and how the chosen parameter variation affects it. So, we establish two different metrics.
The first one is a simple count of the total number of edges present in each of the graphs. Even though it does not give us direct information about the structure, it gives us a general idea of the DAG's complexity, whether it is more densely connected or more sparse. The second metric is based on identifying which variables are connected in one graph but not in the rest, in that way, establishing similarities, when both algorithms have generated the same connection, or differences when a given edge is missing. The comparison is done by iteratively selecting a pair of variables and checking whether the selected path is existent or non-existent in the rest of the graphs.

3 Results

Considering that the goal is to compare the graphs based on their structure, we assume that we do not know any relation between the variables, in order to avoid biasing the comparison.

The resulting networks of the PC algorithm are reported in figures 2 and 3, on Appendix A. The Table 2 containing the results of the evaluation metric, and Table 1 containing the total number of edges per network, can be found in the same section.

Already only taking a glance to the output of this first algorithm we can observe a different structure when alpha (α) is set to a higher (0.05) or lower (0.01) value. α is the significance level for the conditional independence tests. When using a high α we obtain a total number of 13 edges and when using a low α we obtain a total of 10 edges. With a higher alpha we obtain a slightly more densely connected graph, as we would expect, since the higher the significance level is, the more likely is an edge to be included.
Regarding the second metric, looking at the Table 2 we can see that there is a total of 10 connections existing in both DAGs, 2 of the remaining 3 not shared connections are found in $PC_{0.01}$. From this results we can say that the algorithm does not perform that different when setting different values of

[2] https://cran.r-project.org/web/packages/pcalg/pcalg.pdf
[3] https://www.bnlearn.com/documentation/man/hc.html
[4] https://www.bnlearn.com/

alpha, or at least, not when setting a difference of 0.04 between the values of alpha.

The resulting networks of the Hill-climbing algorithm are reported in figures 4 and 5, on Appendix A.

This time, again, it can be appreciated how the DAG created using the AIC scoring function is more densely connected, with a total number of 20 edges. While the DAG using BIC scoring function seems a bit more sparse, with a total number of 15 edges. Looking at the results of the second metric, we can conclude that most of the edges are the same in both networks, with the exception of 4, 3 of which can be found in HC_{AIC} but not in HC_{BIC}, as we would expect, considering that with AIC we obtain a slightly more dense network. We can say that AIC tries to select the model that most adequately describes an unknown observation. This results in a more heavily connected graph as it will be more likely to capture the relationships of a future observation. On the contrary, BIC tries to find the most adequate model that fits the current data.

Now, comparing the results from PC against the results from HC, we observe more distinct results, the different algorithms have learned different structures. In order to account for the variation within the graphs produced by the same algorithm but with a different parameter setting, we rely on averaging the metrics. We can observe that the PC algorithm, in general, produces more sparse networks, less connected, while the HC algorithm tends to have more connections. On average, 57% of the connections exist in the outcome of both algorithms. From the remaining 43% that accounts for all the existing variations, 71% are edges present in HC but not in PC.

Comparing the ground truth DAG (hand-constructed) with the PC algorithm we obtain that on average 41.5% of the connections are shared between them, leaving a large 58.5% that is different. Most of the difference is due to the existence of edges in the original DAG (ground truth) that are not present in the ones obtained with structure learning. For HC we obtain much more similar DAGs, where 73.5% of the edges are shared.

4 Discussion

Just by having a quick look at the produced DAGs it is noticeable that there are some structural differences. Comparing the two algorithms against each other, we can see that a bit more than half of the total existing connections are the same for both results, which leaves a great number of edges that differ. If we only consider the total number of edges from each graph we are not able to tell much about the structure itself, but we can state that HC tends to produce more heavily connected networks than PC, as we would expect, considering that HC starts from an arbitrary solution, and it goes iteratively introducing changes trying to reach for the optimal solution. This workflow is quite similar to the one we used in order to build the ground truth DAG, for which we started from almost a fully-connected graph, and by removing edges we got to what we considered the most optimal solution.
We could say that HC has performed better with our data, than PC, if we consider the manually-constructed graph as the ground truth. While with PC we only obtain 41.5% of similarity, with HC we obtain a 73.5%, proving that it has found a model that better fits the data. As mentioned before, this may be due to the fact that the original DAG has been produced following a similar protocol. The changes on the selected parameter for each algorithm did not result in major changes on the final output, so, we can attribute the exit (in the case of HC) of the good performance, almost fully to the algorithm itself, since for both variations of the same algorithm, the similarity to the ground truth DAG it's practically the same.

Even though HC has given a good solution, it may be still far from optimal. Issues like, getting stuck in a local maxima, prevent the algorithm from finding a better solution, as we can see from the elevated number of edges in the produced graphs, meaning that it could have been the case that the model got stuck in a local maxima, and stopped removing edges. Luckily, some research has been done in order to fix this problems, and algorithms like Local Beam Search or Genetic Algorithms

overcome this kind of issues. A complication that we have encountered is that neither HC nor PC account for latent variables in the data, thus, there is a limit to which the generated model can properly fit the data. Algorithms like RFCI or FCI are being developed in order to include this kind of variables into the structure learning, even though it is a field that still requires a lot of work.

References

[1] Marco Scutari. Learning bayesian networks with the bnlearn r package. *Journal of Statistical Software*, 35(3):1–22, 2010.

[2] Ria Puan Adhitama and Dewi Retno Sari Saputro. Hill climbing algorithm for bayesian network structure. *AIP Conference Proceedings*, 2479(1):020035, 2022.

Appendix

A Figures and Tables

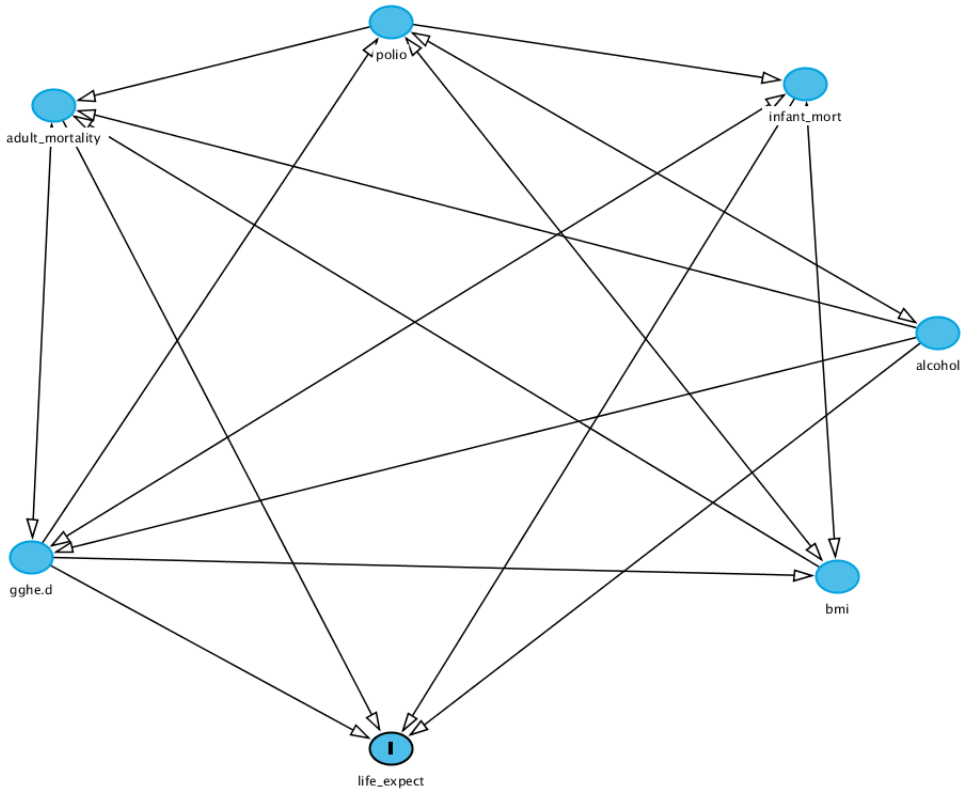

Figure 1: Manually constructed network, considered as "ground truth".

	Total connections
GT	16
$\mathbf{PC}_{0.01}$	10
$\mathbf{PC}_{0.05}$	13
$\mathbf{HC}_{AIC}$	20
$\mathbf{HC}_{BIC}$	15

Table 1: Contains the total number of edges (connections) from each DAG. GT being the ground truth DAG.

	TP	**FP**	**FN**
GT vs. $\mathbf{PC}_{0.01}$	7	8	3
GT vs. $\mathbf{PC}_{0.05}$	9	7	4
GT vs. $\mathbf{HC}_{AIC}$	15	1	5
GT vs. $\mathbf{HC}_{BIC}$	13	3	1
$\mathbf{PC}_{0.01}$ **vs.** $\mathbf{PC}_{0.05}$	10	2	1
$\mathbf{PC}_{0.01}$ **vs.** $\mathbf{HC}_{AIC}$	10	0	9
$\mathbf{PC}_{0.01}$ **vs.** $\mathbf{HC}_{BIC}$	7	2	8
$\mathbf{PC}_{0.05}$ **vs.** $\mathbf{HC}_{AIC}$	13	1	7
$\mathbf{PC}_{0.05}$ **vs.** $\mathbf{HC}_{BIC}$	10	3	5
$\mathbf{HC}_{AIC}$ **vs.** $\mathbf{HC}_{BIC}$	17	3	1

Table 2: Contains the manually counted number of edges of the comparison of two given graphs. TP being the number of edges that appear in both graphs. FP the edges that appear only in the first given graph. And FN the edges that appear only in the second given graph. The ground truth DAG is represented with GT. Each of the algorithms (HC, PC) has specified the corresponding parameter used, alpha (0.01 or 0.05) for PC, and the scoring function (AIC, BIC) for HC.

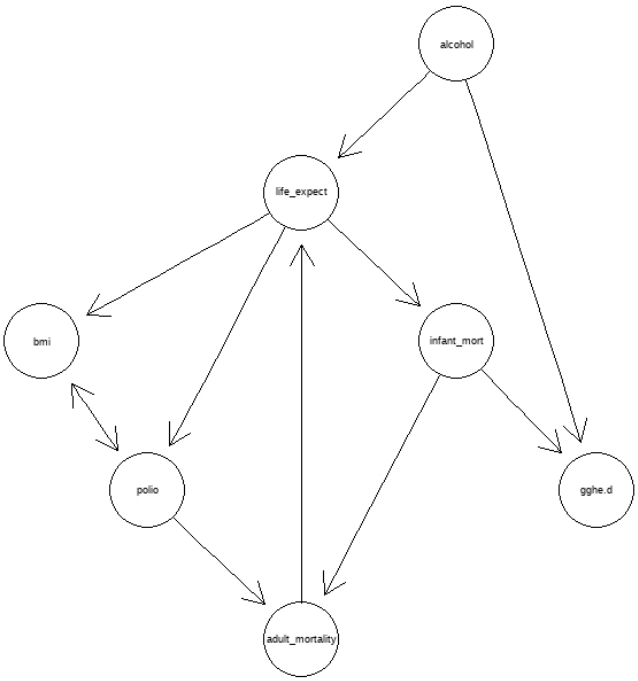

Figure 2: Network obtained with pc algorithm with alpha set to 0.01.

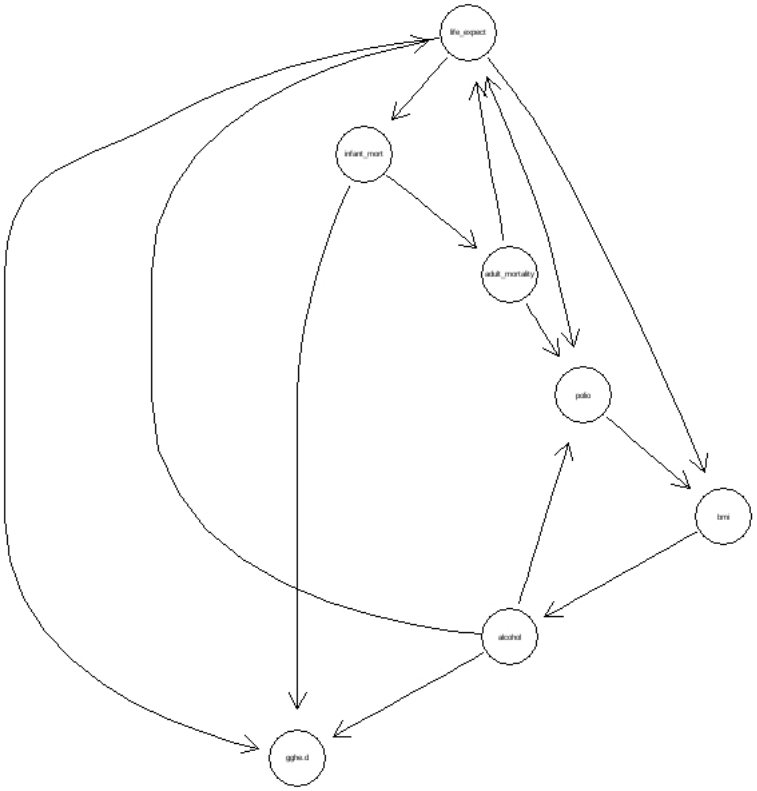

Figure 3: Network obtained with pc algorithm with alpha set to 0.05.

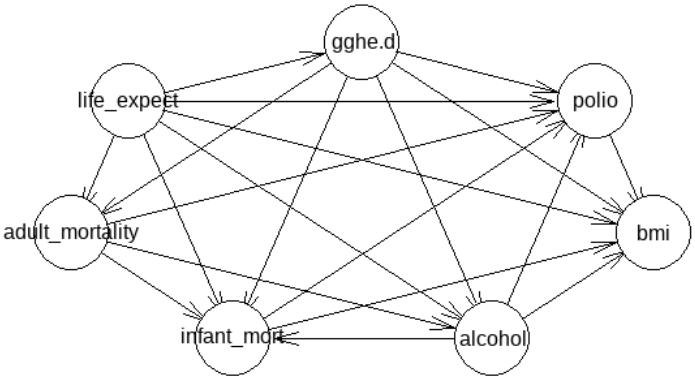

Figure 4: Network obtained with hill climbing algorithm with AIC as score function and 2 restarts.

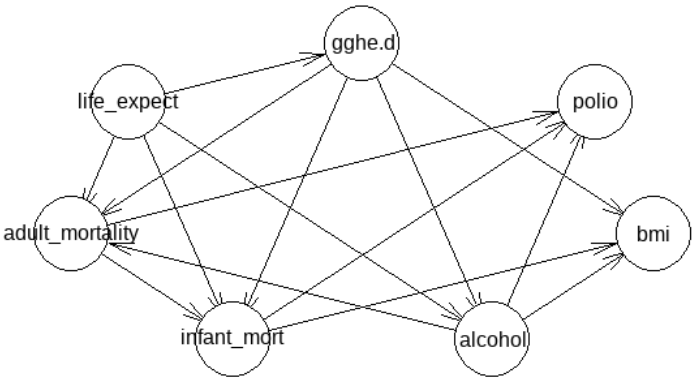

Figure 5: Network obtained with hill climbing algorithm with BIC as score function and 2 restarts.

B R code for PC algorithm

```
#required packages and libraries
install.packages("BiocManager")
BiocManager::install("RBGL")
BiocManager::install("Rgraphviz")
install.packages('pcalg')
library(pcalg)

#load the dataset
lexp <- read.csv("who_life_exp.csv", header = TRUE, stringsAsFactors = TRUE)
str(lexp)

columns <- c("life_expect", "adult_mortality", "infant_mort", "alcohol", "bmi", "
    polio" , "gghe.d")
lexp2 <- lexp[columns]

#just remove the missing values
lexp3 <- na.omit(lexp2)
str(lexp3)
summary(lexp3)

#pc, alpha=0.01
suffStat <- list(C = cor(lexp3), n = nrow(lexp3))
pc.lexp <- pc(suffStat, indepTest = gaussCItest, labels = columns, alpha = 0.01)
plot(pc.lexp)
#pc, alph=0.05
pc.lexp <- pc(suffStat, indepTest = gaussCItest, labels = columns, alpha = 0.05)
plot(pc.lexp)
```

C R code for Hill-climbing algorithm

```
#required packages and libraries
install.packages('bnlearn')
library(bnlearn)

#load the dataset
```

```
lexp <- read.csv("who_life_exp.csv", header = TRUE, stringsAsFactors = TRUE)
str(lexp)

columns <- c("life_expect", "adult_mortality", "infant_mort", "alcohol", "bmi", "
    polio" , "gghe.d")
lexp2 <- lexp[columns]

#just remove the missing values
lexp3 <- na.omit(lexp2)
str(lexp3)
summary(lexp3)

#hc with aic
bn.hc <- hc(lexp3, score='aic-g', restart=2)
bn.hc
plot(bn.hc)
#hc with bic
bn.hc <- hc(lexp3, score='bic-g', restart=2)
bn.hc
plot(bn.hc)
```