Explainability Using Bayesian Networks for Bias Detection: FAIRness with FDO

Ronit Purian[‡,§], Natan Katz[§], Batya Feldman[§]

‡ Tel Aviv University, Tel Aviv, Israel
§ CODATA, Tel Aviv, Israel

Corresponding author: Ronit Purian (purianro@tauex.tau.ac.il)

Abstract

In this paper we aim to provide an implementation of the FAIR Data Points (FDP) spec, that will apply our bias detection algorithm and automatically calculate a FAIRness score (FNS). FAIR metrics would be themselves represented as FDOs, and could be presented via a visual dashboard, and be machine accessible (Mons 2020, Wilkinson et al. 2016). This will enable dataset owners to monitor the level of FAIRness of their data. This is a step forward in making data FAIR, i.e., Findable, Accessible, Interoperable, and Reusable; or simply, Fully AI Ready data.

First we may discuss the context of this topic with respect to Deep Learning (DL) problems. Why are Bayesian Networks (BN, explained below) beneficial for such issues?

- **Explainability** – Obtaining a directed acyclic graph (DAG) from a BN training provides coherent information about independence variables in the data base. In a generic DL problem, features are functions of these variables. Thus, one can derive which variables are dominant in our system. When customers or business units are interested in the cause of a neural net outcome, this DAG structure can be both a source to provide importance and clarify the model.
- **Dimension Reduction** — BN provides the joint distribution of our variables and their associations. The latter may play a role in reducing the features that we induce to the DL engine: If we know that for random variables X,Y the conditional entropy of X in Y are low, we may omit X since Y provides its nearly entire information. We have, therefore, a tool that can statistically exclude redundant variables
- **Tagging Behavior** – This section can be less evident for those who work in domains such as vision or voice. In some frameworks, labeling can be an obscure task (to illustrate, consider a sentiment problem with many categories that may overlap). When we tag the data, we may rely on some features within the datasets and generate conditional probability. Training BN, when we initialize an empty DAG, may provide outcomes in which the target is a parent of other nodes. Observing several tested examples, these outcomes reflect these “taggers’ manners”. We can

therefore use DAGs not merely for the purpose of model development in machine learning but mainly learning taggers policy and improve it if needed.

- **The conjunction of DL and Casual inference** — Causal Inference is a highly developed domain in data analytics. It offers tools to resolve questions that on the one hand, DL models commonly do not and, on the other hand, the real-world raises. There is a need to find a framework in which these tools will work in conjunction. Indeed, such frameworks already exist (e.g., GNN). But a mechanism that merges typical DL problems causality is less common. We believe that the flow, as described in this paper, is a good step in the direction of achieving benefits from this conjunction.
- **Fairness and Bias** – Bayesian networks, in their essence, are not a tool for bias detection but they reveal which of the columns (or which of the data items) is dominant and modify other variables. When we discuss noise and bias, we address these faults to the column and not to the model or to the entire data base. However, assume we have a set of tools to measure bias (Purian et al. 2022). Bayesian networks can provide information about the prominence of these columns (as they are "cause" or "effect" in the data), thus allow us to assess the overall bias in the database.

What are Bayesian Networks?

The motivation for using Bayesian Networks (BN) is to learn the dependencies within a set of random variables. The networks themselves are directed acyclic graphs (DAG), which mimic the joint distribution of the random variables (e.g., Perrier et al. (2008)). The graph structure follows the probabilistic dependencies factorization of the joint distribution: a node V depends only on its parents (a r.v X independent of the other nodes will be presented as a parent free node).

Real-World Example

In this paper we present a way of using the DL engine tabular data, with the python package bnlearn. Since this project is commercial, the variable names were masked; thus, they will have meaningless names.

Constructing Our DAG

We begin by finding our optimal DAG.

```
import bnlearn as bn

DAG = bn.structure_learning.fit(dataframe)
```

We now have a DAG. It has a set of nodes and an adjacency matrix that can be found as follow:

```
print(DAG['adjmat'])
```

The outcome has this form Fig. 1a.

Where rows are sources (namely the direction of the arc is from the left column to the elements in the row) and columns are targets (i.e., the header of the column receives the arcs). When we begin drawing the obtained DAG, we get for one set of variables the following image: Fig. 1b.

We can see that the target node in the rectangle is a source for many nodes. We can see that it still points arrows itself to two nodes. We will discuss this in the discussion (i.e., Rauber 2021). We have more variables, therefore I increased the number of nodes. Adding the information provided a new source for the target (i.e., its entire row is “False”). The obtained graph is the following: Fig. 1c.

So, we know how to construct a DAG. Now we need to train its parameters. Code-wise we perform this as follows:

> model_mle = bn.parameter_learning.fit(DAG, dataframe, methodtype= **'maximumlikelihood'**)

We can change ‘**maximulikelihood**’ with ‘**bayes**’ as described beyond. The outcome of this training is a set of factorized conditional distributions that reflect the DAG’s structure. It has this form for a given variable: Fig. 1d. The code to create DAG presentation is provided in Fig. 2.

Discussion

In this paper we have presented some of the theoretical concepts of Bayesian Networks and the usage they provide in constructing an approximated DAG for a set of variables. In addition, we presented a real-world example of end to end DAG learning: Constructing it using BN, training its parameters using maximum likelihood estimation (MLE) methods, and performing and inference.

FAIR metrics, represented as FDOs, can also be visualised and monitored, taking care of data FAIRness.

Keywords

Bayesian networks (BN), causal inference, conditional entropy, deeg learning, dimension reduction, directed acyclic graph (DAG), neural networks, tagging behavior

Presenting author

Ronit Purian

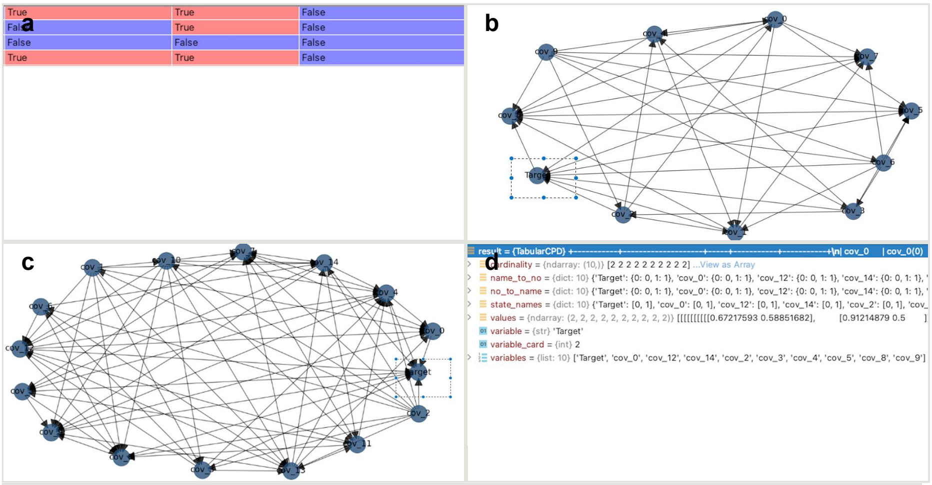
Presented at

First International Conference on FAIR Digital Objects, poster

Conflicts of interest

References

- Mons B (2020) The VODAN IN: support of a FAIR-based infrastructure for COVID-19. European Journal of Human Genetics 28: 724-727. https://doi.org/10.1038/s41431-020-0635-7
- Perrier E, Imoto S, Miyano S (2008) Finding Optimal Bayesian Network Given a Super-Structure. Journal of Machine Learning Research 9 (74): 2251-2286. URL: http://jmlr.org/papers/v9/perrier08a.html
- Purian R, Katz N, Feldman B, Ben-Yosef Y (2022) Unbiased AI. International Data Week (IDW) SciDataCon-IDW (AI & Reproducibility, Repeatability, and Replicability). URL: https://www.scidatacon.org/IDW-2022/sessions/467/paper/1098
- Rauber A (2021) Precisely and Persistently Identifying and Citing Arbitrary Subsets of Dynamic Data. Harvard Data Science Review 3 (4). https://doi.org/10.1162/99608f92.be565013
- Wilkinson M, Dumontier M, Aalbersberg I, Mons B (2016) The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data 3: 160018. https://doi.org/10.1038/sdata.2016.18



True	True	False
False	True	False
False	False	False
True	True	False

```
result = {TabularCPD} +-----------+-------------------+-----+-----------+-----------+\n| cov_0     | cov_0(0)
>  cardinality = {ndarray: (10,)} [2 2 2 2 2 2 2 2 2 2] ...View as Array
>  name_to_no = {dict: 10} {'Target': {0: 0, 1: 1}, 'cov_0': {0: 0, 1: 1}, 'cov_12': {0: 0, 1: 1}, 'cov_14': {0: 0, 1: 1}, '
>  no_to_name = {dict: 10} {'Target': {0: 0, 1: 1}, 'cov_0': {0: 0, 1: 1}, 'cov_12': {0: 0, 1: 1}, 'cov_14': {0: 0, 1: 1}, '
>  state_names = {dict: 10} {'Target': [0, 1], 'cov_0': [0, 1], 'cov_12': [0, 1], 'cov_14': [0, 1], 'cov_2': [0, 1], 'cov_
>  values = {ndarray: (2, 2, 2, 2, 2, 2, 2, 2, 2, 2)} [[[[[[[[[[0.67217593 0.58851682],        [0.91214879 0.5        ]
   variable = {str} 'Target'
   variable_card = {int} 2
>  variables = {list: 10} ['Target', 'cov_0', 'cov_12', 'cov_14', 'cov_2', 'cov_3', 'cov_4', 'cov_5', 'cov_8', 'cov_9']
```

Figure 1.

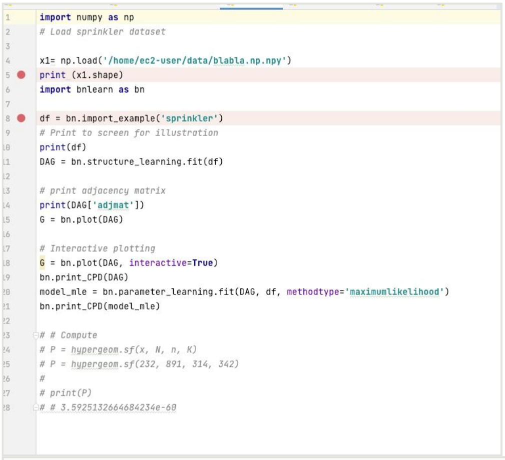
Constructing DAG. Credit: Authors

a: The outcome has this form

b: Target node

c: The obtained graph

d: A set of factorized conditional distributions that reflect the DAG's structure: The training outcome for a given variable



```
import numpy as np
# Load sprinkler dataset

x1= np.load('/home/ec2-user/data/blabla.np.npy')
print (x1.shape)
import bnlearn as bn

df = bn.import_example('sprinkler')
# Print to screen for illustration
print(df)
DAG = bn.structure_learning.fit(df)

# print adjacency matrix
print(DAG['adjmat'])
G = bn.plot(DAG)

# Interactive plotting
G = bn.plot(DAG, interactive=True)
bn.print_CPD(DAG)
model_mle = bn.parameter_learning.fit(DAG, df, methodtype='maximumlikelihood')
bn.print_CPD(model_mle)

# # Compute
# P = hypergeom.sf(x, N, n, K)
# P = hypergeom.sf(232, 891, 314, 342)
#
# print(P)
# # 3.5925132664684234e-60
```

Figure 2.

DAG presentation code. Credit: Authors