Laboratory # 8

Bayesian Networks as a Machine Reasoning Tool

1 Introduction

This laboratory exercise is an introduction to Bayesian networks that offer a causal interpretation of the decision-making process in terms of probability.

The best known tools for calculations on the Bayesian networks include a free MATLAB toolbox BNT[1], developed by Dr. Kevin Murphy (University of British Columbia), as well as the open-source library PyAgrum[2].

In this Lab, we will use PyAgrum.

The default Anaconda installation does not contain the PyAgrum library, thus you must install it manually using the "Anaconda Prompt" as explained in the Pre-Lab and also below:

1. Go to the start menu > Anaconda3 (64-bit) > Anaconda Powershell Prompt.
2. Type: `conda install -c conda-forge pyagrum`
3. NOTE: You may need to uninstall and reinstall the `pydot` package. Notebook, Jupyter and/or Pyagrum might be automatically uninstalled in the process, so please reinstall them again too:
```
conda create -n encm509lab8 python=3.9 numpy matplotlib
conda activate encm509lab8
conda install -c conda-forge pyagrum
conda uninstall pydot
conda install pydot jupyter notebook
conda install -c conda-forge pyagrum
```

2 Bayesian Networks

A Bayesian Network (BN) is a directed acyclic graph derived from a causal network, by adding the Conditional Probability Tables (CPTs). Fig. 1 shows a graphical representation of a BN that will be used later in this Lab. It represents a fragment of a causal graph that describes the infection outbreak on the infamous Diamond Princess Cruise ship in February 2020. The BN shows how age and gender, represented by the parent nodes (`Age` and `Gender`), impact the susceptibility to an infection, represented by the child node `Susceptibility`. The arrows between nodes are directed from the parent to the child node.

In this laboratory, we will use the library PyAgrum to encode a BN and perform inference (computation of posterior probabilities given priors and some evidences). You will use the notebook `Lab08_BayesianNetwork.ipynb`, available on the D2L.

[1] https://github.com/bayesnet/bnt
[2] https://agrum.gitlab.io/

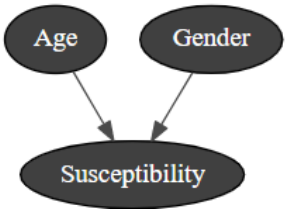

Figure 1: Graphical representation of a BN about the Diamond Princess cruise ship. The arrows show the directional causal relationship between the nodes `Age`, `Gender` and `Susceptibility`.

2.1 Example 1: Winter

Consider a network shown in Fig. 2, represented by 4 nodes/variables:

Winter: The probability that it is a winter season.

Influenza: The probability that a person has influenza.

Cold: The probability that a person has cold.

Fever: The probability that a person has fever.

Each node (variable) is assigned to a table which gives the conditional probabilities for the child node (variable in columns), given the value of the parent node (variables in rows).

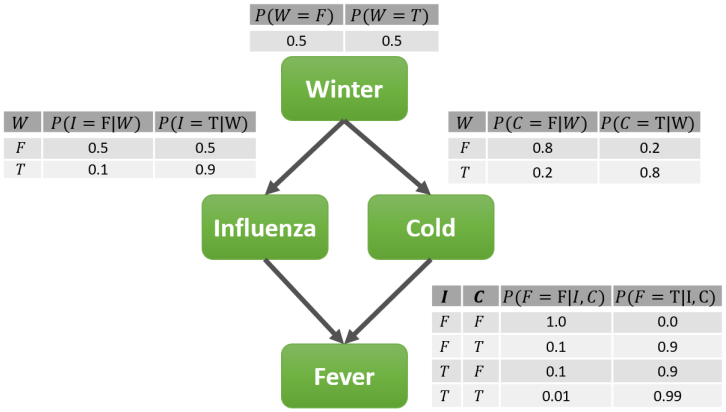

Figure 2: Bayesian Network with 4 nodes.

In PyAgrum, to define the structure of the network, use the following code:

```
# necessary imports from PyAgrum
import pyAgrum as gum
import pyAgrum.lib.notebook as gnb

# definition of the BN structure
bn = gum.fastBN('Winter->Influenza; Winter->Cold; Influenza->Fever<-Cold')
```

In the above code, pay attention to the arrows "`->`" used in the command `gum.fastBN(...)`, as they define the causal relationship between the nodes. The CPTs are randomly defined. To access the CPTs created for each node, run the following command:

```
# Showing the BN with the CPTs
gnb.showInference(bn)
```

The BN shown should be *similar* to the one in Fig. 3. Remember that all the values have been randomly assigned so far.

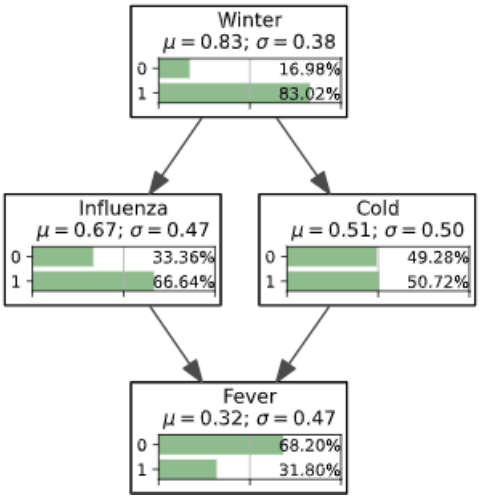

Figure 3: A BN with 4 nodes and the CPTs randomly assigned.

Now, the CPTs must be filled in with the real data (see the CPTs in Fig. 2). In PyAgrum, to rewrite a CPT's data, use the command: `bn.cpt('NODE NAME')[:] = [ values here ]`. See the comments in the code below for clarification regarding the position of each value.

```
# P(Winter = False) = 0.5
# P(Winter = True) = 0.5
bn.cpt('Winter')[:] = [0.5, 0.5]

# P(Cold = False | Winter = False) = 0.8
# P(Cold = True | Winter = False) = 0.2
bn.cpt('Cold')[{'Winter':0}] = [0.8, 0.2]

# P(Cold = False | Winter = True) = 0.2
# P(Cold = True | Winter = True) = 0.8
bn.cpt('Cold')[{'Winter':1}] = [0.2, 0.8]

bn.cpt('Influenza')[{'Winter':0}] = [0.5, 0.5]
bn.cpt('Influenza')[{'Winter':1}] = [0.1, 0.9]

# P(Fever = False | Influenza = False, Cold = False) = 1.0
# P(Fever = True | Influenza = False, Cold = False) = 0.0
bn.cpt('Fever')[{'Influenza':0, 'Cold':0}] = [1.0, 0.0]
bn.cpt('Fever')[{'Influenza':0, 'Cold':1}] = [0.1, 0.9]
bn.cpt('Fever')[{'Influenza':1, 'Cold':0}] = [0.1, 0.9]
bn.cpt('Fever')[{'Influenza':1, 'Cold':1}] = [0.01, 0.99]
```

```
# show the BN with the CPTs updated
gnb.showInference(bn)
```

After updating the CPTs, the BN will look like the one shown in Fig. 4.

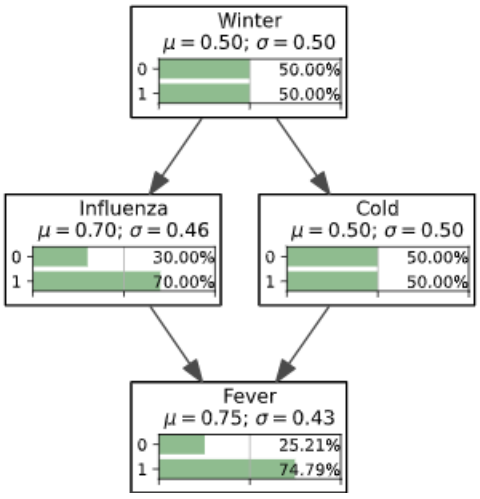

Figure 4: A BN with the CPTs updated using real data.

2.1.1 Inference in the network

Having created the BN, we can now use update the belief (posterior probability), given an evidence and a prior probability it for inference. The *Lazy Propagation* algorithm will be used to perform the inference. Alternatively, in PyAgrum, the algorithm *Variable Elimination* can also be used.

Consider the example below, where we define the evidence of Fever to be True: $P(Fever = True)$.

```
# evidence of Fever is True (1)
evidence = {'Fever': True}

# in PyAgrum two algorithms are available to do inference:
# — Lazy Propagation (default): gum.LazyPropagation(bn)
# — Variable Elimination: gum.VariableElimination(bn)
ie = gum.LazyPropagation(bn)

ie.setEvidence(evidence)
ie.makeInference()

# getting the posterior probabilities for Influenza
res_influenza = ie.posterior('Influenza')
```

The result, stored in variable `res_influenza`, will be an array of probabilities for Influenza. To narrow the values down, use the index value 0 for False and 1 for True:

```
# calculating the probability that somebody has Influenza considering that
# they already have Fever
print('P(Influenza = True | Fever = True): %.4f%%' % (res_influenza[1] * 100))
```

```
Out: P(Influenza = True | Fever = True): 89.1697%
```

will return the value $P\left(Influenza = True | Fever = True\right) = 89.17\%$.

2.1.2 More Inference

Let us add evidence for the node Cold as well:

```
# adding a new evidence: Cold = True
evidence = {'Fever': True, 'Cold': True}

ie = gum.LazyPropagation(bn)

ie.setEvidence(evidence)
ie.makeInference()

res_influenza = ie.posterior('Influenza')
```

As seen in Section 2.1.1, after execution of the command `ie.posterior(...)`, the variable `res_influenza` gets the values (probabilities) for $P\left(Influenza | Cold = True, Fever = True\right)$. Thus, the probability that somebody has Influenza considering that they have Fever and Cold is assessed as follows:

```
res_influenza = ie.posterior('Influenza')
print('P(Influenza = False | Fever = True, Cold = True): %.4f%%' % \
            (res_influenza[0] * 100))

Out: P(Influenza = False | Fever = True, Cold = True): 16.6359%
```

returns the result of $P\left(Influenza = False | Fever = True, Cold = True\right) = 16.63\%$.

2.1.3 Computing Joint Probability

The joint probability is computed given the assignment for all or a subset network's variables; it allows for assessing the likelihood of a certain scenario. For example, we can calculate the joint probability of the event given the assignment for $Influenza$, $Cold$ and $Fever$:

```
# Calculating the joint probability: Chain Rule
pICF = bn1.cpt('Influenza') * bn1.cpt('Cold') * bn1.cpt('Fever') * bn1.cpt('Winter')
```

If one or more variable assignment are not specified, it is marginalized:

```
# Marginalizing Winter
pICF = pICF.margSumOut(['Winter'])
pICF.normalize()
```

The variable `pICF` has the dimensions $2 \times 2 \times 2$, where each dimension corresponds to one of the variables, Fever, Influenza and Cold. We can query some scenarios from the network, for example:

What is the probability of someone having a fever in the absence of influenza or cold?

```
# the assignment 1,0,0 in pICF corresponds to Fever=1, Influenza=0 and Cold=0
print('P(Fever = T, Influenza = F, Cold = F): %.4f%%' % (pICF[1,0,0] * 100))

Out: P(Fever = T, Influenza = F, Cold = F): 0.0000%
```

Note that $P\,(Fever = T, Influenza = F, Cold = F) = 0$, since we assume no Fever if both Influenza and Cold are false.

2.2 Example 2: Bayesian Network of Diamond Princess cruise ship

The Diamond Princess (DP) cruise ship is infamously known for the first massive COVID-19 outbreak on board; it happened in the beginning of 2020[3].

In this exercise, we are going to use the DP demographic data (used in our first Lab!) as shown below:

```
# % of male and female
gender = [55, 45]

# stratified passengers' age data from the agency's report, 2020
age = [16, 23, 347, 428, 334, 398, 923, 1015, 216, 11]
```

2.2.1 Creating the network and setting the CPTs

Based on the data shown below, we created a BN structure using the following command:

```
bn = gum.fastBN('Age{0—9|10—19|20—29|30—39|40—49|50—59' +
                '|60—69|70—79|80—89|90—99}—>Susceptibility{low|medium|high};' +
                'Gender{male|female}—>Susceptibility;')
bn.cpt('Age')[:] = age / np.sum(age)
bn.cpt('Gender')[:] = gender / np.sum(gender)
# 0—9 yo, male: Susceptibility is 95% "low"
bn.cpt('Susceptibility')[{'Age':0, 'Gender':0}] = [.95, .05, .00]
# 0—9 yo, female: Susceptibility is 99% "low"
bn.cpt('Susceptibility')[{'Age':0, 'Gender':1}] = [.99, .01, .00]
# 90—99 yo, male: Susceptibility is 99% "high"
bn.cpt('Susceptibility')[{'Age':9, 'Gender':0}] = [.00, .01, .99]
# 90—99 yo, female: Susceptibility is 98% "high"
bn.cpt('Susceptibility')[{'Age':9, 'Gender':1}] = [.00, .02, .98]
# 40—49 yo, male: Susceptibility is 50% "high"
bn.cpt('Susceptibility')[{'Age':4, 'Gender':0}] = [.20, .30, .50]
# 40—49 yo, female: Susceptibility is 40% "medium"
bn.cpt('Susceptibility')[{'Age':4, 'Gender':1}] = [.30, .40, .30]
gnb.showInference(bn, size='30')
```

The node *age* has 10 states, and the node *susceptibility* has 3 states (Fig. 5).

[3] https://en.wikipedia.org/wiki/COVID-19_pandemic_on_Diamond_Princess

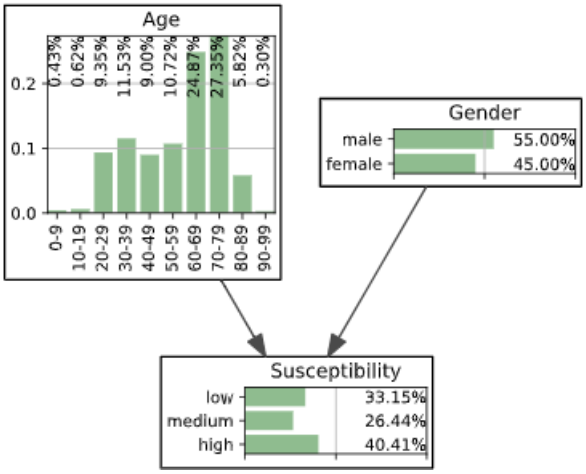

Figure 5: Bayesian Network with Diamond Princess cruise ship data.

2.2.2 Inference

With the network created, we can perform an inference using the same commands as in the previous example. Let us consider an evidence that a subject is *male*. How susceptible that person is to the infection, and what is the most probable age for that individual ?

```
# considering only  male passenderg
evidence = {'Gender':0}

ie = gum.LazyPropagation(bn)

ie.setEvidence(evidence)
ie.makeInference()

# getting the posterior CPTs
res_age = ie.posterior('Age')
res_susc = ie.posterior('Susceptibility')

res_age_a = res_age.toarray() * 100
res_suc_a = res_susc.toarray() * 100

max_idx = np.argmax(res_age_a)
max_perc = res_age_a[max_idx]
max_age = age_labels[max_idx]

print('High susceptibility of %.2f%%, the age [%s] is the most likely with %.2f%%' %\
            (res_suc_a[2], max_age, max_perc))

Out: Susceptibility of 39.89\%, and the age [70—79] is the most likely with 27.35%
```

Thus, based on the data and the analysis of the network output:

- $P(Susceptibility = High|Gender = Male) = 39.89\%$

- $P(Age = 70 - 79 | Gender = Male) = 27.35\%$

3 Lab Report

Your report in the form of a Jupyter Notebook/Python (file extension `.ipynb`) shall include the description of each exercise with illustrations/graphs and analysis of the results (marks are distributed as shown in the Exercise section, 10 marks total). Save your Notebook using menu "Download As" as `.ipynb`, and submit to D2L dropbox for Lab 8, by the end of the day on the following Thursday.

4 Lab Exercise in Jupyter Notebook with Python

- **Exercise 1** (3 marks): Consider Section 2.1.1 of Example 1. Perform calculations, by hand, of the probabilities that Influenza is True, and then that it is False, given the evidence that Fever is True. Compare the results to the ones calculated by using PyAgrum.
- **Exercise 2** (3 marks): In Section 2.1.3, PyAgrum was used to calculate the joint probability of Influenza, Cold and Fever. Repeat the same experiment for Fever, Cold and Winter. Next, do the calculations manually and compare to PyAgrum's results.
- **Exercise 3** (1 marks): In the CPTs for the BN of the Diamond Princess (Section 2.2.1), not all the entries of Susceptibility node were defined. Explain how many entries should be defined and what is the impact of having an incomplete CPT.
- **Exercise 4** (3 marks): Using the BN of the Diamond Princess case, what will be the most likely the *Age* group of an individual given that this individual is a male with 'high' *Susceptibility*? Do the calculations manually for cases specified by the table, such as for bins 0, 4 and 9, and then compare against the Python's results.

5 Acknowledgments

The PyAgrum is a free Python library available at https://agrum.gitlab.io/. The data for the DP cruise example was prepared by Dr. H. C. R. Oliveira. We also acknowledge this course TAs, I. Yankovyi and M. Zakir, for verifying this lab code.

Svetlana Yanushkevich
March 10, 2023