



Subject: Probabilistic Graph Models (DJ19DSC402)

AY: 2022-23

Bhuvi Ghosh
60009210191

Experiment 2

Aim: Implement Monty Hall Problem using Bayesian Networks

Theory:

- Let's assume we have three doors with different choices, one door has ice, one has water and last has fire, which one should we select based on a certain situation. The selection of the switching of choice is done using the Bayesian network. Using the Bayesian network, we have to understand if we should switch our choice or not. We represent this Monty Hall problem using Directed Acyclic Graph (DAG).
- If I pick a door and hold it, I have a $1/3$ chance of winning.

My first guess is 1 in 3 — there are 3 random options, right?

If I rigidly stick with my first choice no matter what, I can't improve my chances. Monty could add 50 doors, blow the other ones up, do a voodoo rain dance — it doesn't matter. The best I can do with my original choice is 1 in 3. The other door must have the rest of the chances, or $2/3$.

The explanation may make sense, but doesn't explain *why* the odds "get better" on the other side. (Several readers have left their own explanations in the comments — try them out if the $1/3$ stay vs $2/3$ switch doesn't click).

- The Monty Hall problem can be considered similar to a statistical illusion. The statistical illusion occurs because your brain's process for evaluating probabilities in the Monty Hall problem is based on a false assumption. Similar to optical illusions, the illusion can seem more real than the actual answer.
- According to Bayes Theorem, we can determine probabilities of "What is probability of an item behind door 1, given that the host had opened door 3?", here A is the event where the host opens door 3.

$$P(E_1|A) = \frac{P(A|E_1).P(E_1)}{P(A|E_1).P(E_1) + P(A|E_2).P(E_2) + P(A|E_3).P(E_3)}$$

We get the probability of the chances an item is behind the door.

Lab Assignments to complete in this session

Use the given dataset and perform the following tasks:


Dataset 1: Set of discrete probabilities or sprinkler dataset

```
import bnlearn as bn
df = bn.import_example('sprinkler') #sprinkler dataset
```

1. Create DAG of the bayesian network for this data
2. Create a predictive model and predict the probabilities for different cases.

For example: you initially said door A, that Monty then opened door B, but that the actual item was behind door C. 6 possible cases:

https://www.researchgate.net/publication/229766475_The_Monty_Hall_Three_Doors_Problem

68  `!pip install pgmpy`

 Collecting pgmpy

Downloading pgmpy-0.1.23-py3-none-any.whl (1.9 MB)


1.9/1.9 MB 19.5 MB/s eta 0:00:00


Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from pgmpy) (3.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from pgmpy) (1.23.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from pgmpy) (1.11.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from pgmpy) (1.2.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from pgmpy) (1.5.3)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.10/dist-packages (from pgmpy) (3.1.1)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (from pgmpy) (2.0.1+cu118)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/dist-packages (from pgmpy) (0.14.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from pgmpy) (4.66.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from pgmpy) (1.3.2)
Requirement already satisfied: opt-einsum in /usr/local/lib/python3.10/dist-packages (from pgmpy) (3.3.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->pgmpy) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->pgmpy) (2023.3.post1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->pgmpy) (3.2.0)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-packages (from statsmodels->pgmpy) (0.5.3)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels->pgmpy) (23.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch->pgmpy) (3.12.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from torch->pgmpy) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch->pgmpy) (1.12)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch->pgmpy) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages (from torch->pgmpy) (2.0.0)
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch->pgmpy) (3.27.4.1)
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch->pgmpy) (16.0.6)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.2->statsmodels->pgmpy) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch->pgmpy) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch->pgmpy) (1.3.0)
Installing collected packages: pgmpy
Successfully installed pgmpy-0.1.23

08 [7] `from pgmpy.models import BayesianNetwork`
`from pgmpy.factors.discrete import TabularCPD`
`model=BayesianNetwork([("C", "H"), ("P", "H")])`
`cpd_c=TabularCPD("C", 3, [[0.33], [0.33], [0.33]])`
`cpd_p=TabularCPD("P", 3, [[0.33], [0.33], [0.33]])`
`cpd_h=TabularCPD("H",`
`3,`
`[[0,0,0,0.5,1,0,1,0.5],`
`[0.5,0,1,0,0,0,1,0,0.5],`
`[0.5,1,0,1,0.5,0,0,0,0]],`
`evidence=[("C", "P"),`
`evidence_card=[3,3])`
`model.add_cpds(cpd_c, cpd_p, cpd_h)`
`model.get_cpds()`
`model.check_model()`

True

08 [9] `from pgmpy.inference import VariableElimination`
`infer=VariableElimination(model)`
`posterior_probability=infer.query([("P"), evidence={"C":0, "H":1}])`

08  `print(posterior_probability)`




P	phi(P)
P(0)	0.3333
P(1)	0.0000
P(2)	0.6667

✓
0s [10]

P	phi(P)
P(0)	0.3333
P(1)	0.0000
P(2)	0.6667

✓
0s [11] infer=VariableElimination(model)
posterior_probability=infer.query(["P"],evidence={"C":0,"H":2})

✓
0s  print(posterior_probability)

P	phi(P)
P(0)	0.3333
P(1)	0.6667
P(2)	0.0000