

POPL Milestone 2 by

Jatin Dahiya - 2021A7PS2067G –

f20212067@goa.bits-pilani.ac.in

Pratyush Suvarna - 2021A7PS1441G -

f20211441@goa.bits-pilani.ac.in

CONCLUSIONS AT THE END.

Implementations completed:


- Used PGMPY Library in python to use Bayesian Networks
- Implemented a specific Bayesian Network in Figaro for calculations for an allergy case.
- Analysed various health conditions in Figaro and Python using **Bayesian Networks**, thereby understanding causal effects and how presence of **evidence changes the probability of a cause**.
- The cases analysed include **Lung Cancer** and **Basic Allergies**.

Comparisons Made:

- **Reliability and Length:** Python Code even after so much abstraction and using complex data structures, is longer and difficult to understand, whereas **Figaro Code only contains random variables and is simpler to understand**.
- **Time of Execution:** Python code takes approximately **25% longer** to execute where Figaro takes 4.5 seconds and Python takes 6 second to execute.
- **Memory for Execution:** The python and Figaro code seem to **use the same amount of memory** and not much difference in that aspect.

Allergy Code for Bayesian Networks

Figaro Code

code-orig > Rash > Figaro >  allergy.scala

```
1  import com.cra.figaro.language._
2  import com.cra.figaro.library.compound._
3  import com.cra.figaro.algorithm.factored.VariableElimination
4
5  object AllergyInference {
6    def main(args: Array[String]): Unit = {
7      Universe.createNew()
8
9      val highPollen = Flip(0.7)
10     val ateAllergicFood = Flip(0.9)
11
12     val hasAllergy = Chain(highPollen, ateAllergicFood, (pollen: Boolean, food: Boolean) =>
13       if (pollen || food) Select(0.9 -> true, 0.1 -> false)
14       else Select(0.5 -> true, 0.5 -> false))
15
16     val sneeze = If(hasAllergy, Flip(0.7), Flip(0.3))
17     val rash = If(hasAllergy, Flip(0.9), Flip(0.1))
18
19     val algorithm = VariableElimination(hasAllergy)
20
21     sneeze.observe(true)
22     rash.observe(true)
23
24     algorithm.start()
25
26     val probabilityOfAllergy = algorithm.probability(hasAllergy, true)
27
28     println("Probability of having an allergy given sneezing and rash: " + probabilityOfAllergy)
29
30     algorithm.kill()
31   }
32 }
33
```

Multiple main classes detected, select one to run:

- [1] AllergyInference
- [2] LungCancerInference

Enter number: 1

[info] Running AllergyInference

Probability of having an allergy given sneezing and rash: 0.9940298507462686

[success] Total time: 9 s, completed 20-Nov-2023, 6:57:48 pm

> |

Python Code:

```
code-orig > Rash > PythonCode > allergy.py > ...
1 # Now defining the parameters.
2 from pgmpy.models import BayesianNetwork
3 from pgmpy.factors.discrete import TabularCPD
4 from pgmpy.inference import VariableElimination
5
6
7 Allergy_model = BayesianNetwork([
8     ('Pollen', 'Allergy'),
9     ('AteAllergicFood', 'Allergy'),
10    ('Allergy', 'Sneeze'),
11    ('Allergy', 'Rash')
12 ])
13 cpd_poll = TabularCPD(variable='Pollen', variable_card=2, values=[
14     [0.7], [0.3]], state_names={'Pollen': ['High', 'Low']})
15 cpd_allfood = TabularCPD(variable='AteAllergicFood', variable_card=2, values=[
16     [0.9], [0.1]], state_names={'AteAllergicFood': ['True', 'False']})
17 cpd_Allergy = TabularCPD(
18     variable='Allergy',
19     variable_card=2,
20     values=[
21         [0.9, 0.9, 0.9, 0.5],
22         [0.1, 0.1, 0.1, 0.5]
23     ],
24     evidence=['Pollen', 'AteAllergicFood'],
25     evidence_card=[2, 2],
26     state_names={
27         'Allergy': ['True', 'False'],
28         'Pollen': ['High', 'Low'],
29         'AteAllergicFood': ['True', 'False']
30     }
31 )
32 cpd_Sneeze = TabularCPD(
33     variable='Sneeze',
34     variable_card=2,
35     values=[
36         [0.7, 0.3],
37         [0.3, 0.7]
38     ],
39     evidence=['Allergy'],
40     evidence_card=[2],
41     state_names={
42         'Sneeze': ['True', 'False'],
43         'Allergy': ['True', 'False']
44     }
45 )
46
47 cpd_rash = TabularCPD(
48     variable='Rash',
49     variable_card=2,
50     values=[
51         [0.9, 0.1],
52         [0.1, 0.9]
53     ],
54     evidence=['Allergy'],
55     evidence_card=[2],
56     state_names={
57         'Rash': ['True', 'False'],
58         'Allergy': ['True', 'False']
59     }
60 )
61
62 # Associating the parameters with the model structure.
63 Allergy_model.add_cpds(cpd_poll, cpd_allfood,
64                        cpd_Allergy, cpd_Sneeze, cpd_rash)
65
66 # Checking if the cpds are valid for the model.
67 assert Allergy_model.check_model()
68
69
70 infer = VariableElimination(Allergy_model)
71
72 print(infer.query(['Allergy'], evidence={
73     'Sneeze': 'True',
74     'Rash': 'True'
75 })))
76
77
78 PS E:\BitsGoa\Sem 5\Principle Of ProLang\BasyesianNetworks> & C:\Users\Jatin\AppData\Local\Programs\Python\Python311\python.exe "e:\BitsGoa\Sem 5\Principle Of
79 ProLang\BasyesianNetworks\code-orig\Rash\PythonCode\allergy.py"
+-----+-----+
| Allergy | phi(Allergy) |
+-----+-----+
| Allergy(True) | 0.9940 |
+-----+-----+
| Allergy(False) | 0.0060 |
+-----+-----+
PS E:\BitsGoa\Sem 5\Principle Of ProLang\BasyesianNetworks> 
```

Cancer Code for Bayesian Networks

Figaro Code

```
code-orig > LungCancer > Figaro > lungCancer.scala
1  import com.cra.figaro.language._
2  import com.cra.figaro.library.compound._
3  import com.cra.figaro.algorithm.factored.VariableElimination
4
5  object LungCancerInference {
6    def main(args: Array[String]): Unit = {
7      Universe.createNew()
8
9      val pollution = Flip(0.1)
10     val smoker = Flip(0.3)
11
12     val lungCancer = CPD(pollution, smoker,
13       (false, false) -> Flip(0.001),
14       (false, true) -> Flip(0.03),
15       (true, false) -> Flip(0.02),
16       (true, true) -> Flip(0.05)
17     )
18
19     val xRay = If(lungCancer, Flip(0.9), Flip(0.2))
20     val dyspnoea = If(lungCancer, Flip(0.65), Flip(0.30))
21
22     val algorithm = VariableElimination(lungCancer)
23
24     xRay.observe(true)
25     dyspnoea.observe(true)
26
27     algorithm.start()
28
29     val poobabilityOfCancer = algorithm.probability(lungCancer, true)
30
31     println("Probability of having lung cancer given xRay and dyspnoea: " + poobabilityOfCancer)
32
33     algorithm.kill()
34   }
35 }
36
```

Multiple main classes detected, select one to run:

- [1] AllergyInference
- [2] LungCancerInference

Enter number: 2

[info] Running LungCancerInference

Probability of having lung cancer given xRay and dyspnoea: 0.10291918630376327

[success] Total time: 2 s, completed 20-Nov-2023, 7:04:10 pm

Python Code:

```
1 # Now defining the parameters.
2 from pgmpy.models import BayesianNetwork
3 from pgmpy.factors.discrete import TabularCPD
4 from pgmpy.inference import VariableElimination
5
6
7 cancer_model = BayesianNetwork([
8     ('Pollution', 'Cancer'),
9     ('Smoker', 'Cancer'),
10    ('Cancer', 'XRay'),
11    ('Cancer', 'Dyspnoea')
12 ])
13 cpd_poll = TabularCPD(variable='Pollution', variable_card=2, values=[
14     [0.1], [0.9]], state_names={'Pollution': ['High', 'Low']})
15 cpd_smoke = TabularCPD(variable='Smoker', variable_card=2, values=[
16     [0.3], [0.7]], state_names={'Smoker': ['True', 'False']})
17 cpd_cancer = TabularCPD(
18     variable='Cancer',
19     variable_card=2,
20     values=[
21         [0.05, 0.02, 0.03, 0.001],
22         [0.95, 0.98, 0.97, 0.999]
23     ],
24     evidence=[('Pollution', 'Smoker')],
25     evidence_card=[2, 2],
26     state_names={
27         'Cancer': ['True', 'False'],
28         'Pollution': ['High', 'Low'],
29         'Smoker': ['True', 'False']
30     }
31 )
32 cpd_xray = TabularCPD(
33     variable='XRay',
34     variable_card=2,
35     values=[
36         [0.9, 0.2],
37         [0.1, 0.8]
38     ],
39     evidence=[('Cancer')],
40     evidence_card=[2],
41     state_names={
42         'XRay': ['True', 'False'],
43         'Cancer': ['True', 'False']
44     }
45 )
46 cpd_dysp = TabularCPD(
47     variable='Dyspnoea',
48     variable_card=2,
49     values=[
50         [0.65, 0.3],
51         [0.35, 0.7]
52     ],
53     evidence=[('Cancer')],
54     evidence_card=[2],
55     state_names={
56         'Dyspnoea': ['True', 'False'],
57         'Cancer': ['True', 'False']
58     }
59 )
60
61
62 # Associating the parameters with the model structure.
63 cancer_model.add_cpds(cpd_poll, cpd_smoke, cpd_cancer, cpd_xray, cpd_dysp)
64
65 # Checking if the cpds are valid for the model.
66 assert cancer_model.check_model()
67
68
69 infer = VariableElimination(cancer_model)
70
71 print(infer.query(['Cancer'], evidence={
72     'XRay': 'True',
73     'Dyspnoea': 'True'
74 })))
75
```

```
PS E:\BitsGoa\Sem 5\Principle Of ProLang\BasyesianNetworks> & C:/Users/Jatin/AppData/Local/Programs/Python/Python311/python.exe "e:/BitsGoa/Sem 5/Principle Of ProLang/BasyesianNetworks/code-orig/LungCancer/PythonCode/lungCancer.py"
```

```
+-----+
| Cancer | phi(Cancer) |
+-----+
| Cancer(True) | 0.1029 |
+-----+
| Cancer(False) | 0.8971 |
+-----+
```

```
PS E:\BitsGoa\Sem 5\Principle Of ProLang\BasyesianNetworks> 
```

Conclusions:

Figaro code seems to run faster and is very easy to comprehend as a programmer. Code **seems easy to update in future iterations** as well due to automatic internal calculations that make updating the code less tedious.

Figaro Code vs Python Code:

- Figaro Code is shorter.
- Figaro Code is faster.
- Both use approximately the same memory.
- Easier comprehension in Figaro.

Bayesian Networks in Figaro can be efficiently used to study causal effects and further research on other health matters and effects. Cancer and Allergy reports are used as examples.

Future Work Possibilities:

We have implemented Bayesian Networks for certain cases, but a general Bayesian Network using OOPS can be implemented in Figaro to make specific implementations even easier and provide a platform to further study causal effects in data, forming Bayesian networks from data using various techniques.