

AIM: To Implement Inferencing with Bayesian Network in python

What is a Bayesian Network?

A Bayesian Network (also called a *Belief Network*) is a type of Probabilistic Graphical Model (PGM). It represents variables and their relationships using probability and graph theory. These networks are widely used to predict outcomes under uncertainty, based on observed evidence.

It is especially useful when we want to make decisions or predictions based on incomplete data — for example, diagnosing a disease based on symptoms or predicting a student's performance.

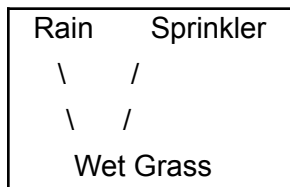
Key Concepts:

1. Directed Acyclic Graph (DAG)

A Bayesian Network is structured as a Directed Acyclic Graph (DAG). In this graph:

- Nodes represent random variables.
- Directed edges (arrows) show conditional dependencies between variables.
- Acyclic means no variable can loop back to influence itself.

Include this diagram in your document:



This is a popular example where we predict if the grass is wet depending on whether it rained or the sprinkler was on.

2. Conditional Probability

The Conditional Probability of an event A given B is written as:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

This means the probability of A occurring if we already know B has happened.

In a Bayesian Network, each node has a Conditional Probability Table (CPT) that defines the probability of that variable given its parent variables.

3. Joint Probability

The Joint Probability is the probability of all variables occurring together. For a Bayesian Network with variables X_1, X_2, \dots, X_n , the joint probability is calculated as:

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i | \text{Parents}(X_i))$$

This formula says: "The overall probability is the product of the conditional probabilities of each node given its parents."

Example Use Cases:

Bayesian Networks are used in many real-world scenarios:

- Medical Diagnosis: Predicting the probability of a disease based on symptoms.
- Student Performance: Estimating admission chances based on IQ, exam level, and marks.
- AI Decision Systems: Making intelligent choices in uncertain conditions.
- Fault Detection: In industries like aircraft or nuclear plants.

How Inferencing Works

Inferencing means calculating the probability of a variable based on known values of other variables. For example:

- What is the chance a student gets admitted given that their marks are high?
- What is the probability it rained if the grass is wet?

These queries are answered using inference algorithms like Variable Elimination, which are built into Python libraries like [pgmpy](#).

```
!pip install pgmpy
```

```
from pgmpy.models import DiscreteBayesianNetwork
from pgmpy.factors.discrete import TabularCPD
from pgmpy.inference import VariableElimination
```

```
# Create the structure of the network
```

```
model = DiscreteBayesianNetwork([
    ('Rain', 'WetGrass'),
    ('Sprinkler', 'WetGrass'),
    ('Rain', 'Sprinkler')
])
```

```
# CPD for Rain
```

```
cpd_rain = TabularCPD(variable='Rain', variable_card=2, values=[[0.7],
[0.3]])
```

```
# CPD for Sprinkler (depends on Rain)
```

```
cpd_sprinkler = TabularCPD(
    variable='Sprinkler', variable_card=2,
    values=[[0.4, 0.9], # Sprinkler = False
            [0.6, 0.1]], # Sprinkler = True
    evidence=['Rain'],
    evidence_card=[2]
)
```

```
# CPD for WetGrass (depends on both Rain and Sprinkler)
```

```
cpd_wetgrass = TabularCPD(
    variable='WetGrass', variable_card=2,
    values=[
        [1.0, 0.1, 0.1, 0.01], # WetGrass = False
        [0.0, 0.9, 0.9, 0.99]  # WetGrass = True
    ],
    evidence=['Sprinkler', 'Rain'],
    evidence_card=[2, 2]
)
```

```
# Add CPDs to the model
```

```
model.add_cpds(cpd_rain, cpd_sprinkler, cpd_wetgrass)
```

```
# Check if the model is valid
```

```
assert model.check_model()
```

```
print("Model is valid!")
```



Model is valid!


```
# Perform inference
```

```
infer = VariableElimination(model)
```

```
# Query: What is the probability that it's raining given that the grass is wet?
```

```
result = infer.query(variables=['Rain'], evidence={'WetGrass': 1})
```

```
print(result)
```




Rain	phi(Rain)
Rain(0)	0.5809
Rain(1)	0.4191

```
# Query: What is the probability of Sprinkler being ON given WetGrass is True?
```

```
result2 = infer.query(variables=['Sprinkler'], evidence={'WetGrass': 1})
```

```
print(result2)
```



Sprinkler	phi(Sprinkler)
Sprinkler(0)	0.3734
Sprinkler(1)	0.6266

Conclusion :

Bayesian Networks are powerful tools that combine graph theory and probability to model uncertainty in real-world problems. They help in making logical predictions even when some data is missing. The ability to represent and reason under uncertainty makes them widely used in AI, medicine, education, and engineering systems.

[AI&DS Expt 01](#)