Uncertain Knowledge and Reasoning with Probabilities

In real-world scenarios, knowledge is often incomplete, uncertain, or noisy. Traditional logic-based reasoning (e.g., Boolean logic) struggles with uncertainty because it assumes true or false values. To handle this, AI and computer systems use probabilistic reasoning to model uncertainty and make decisions based on incomplete information.

1. Uncertainty in AI and Knowledge Representation

Uncertainty arises due to:

- Incomplete data (e.g., missing medical records).
- Ambiguity (e.g., "It may rain tomorrow").
- Noisy data (e.g., sensor errors in robotics).
- Conflicting evidence (e.g., different doctors giving different diagnoses).

To handle these uncertainties, we use probability theory and Bayesian reasoning.

2. Probabilistic Reasoning Methods

A. Bayesian Networks (Belief Networks)

A Bayesian Network (BN) is a graphical model representing random variables and their probabilistic dependencies using conditional probabilities.

Example: Medical Diagnosis

Let's say we want to determine if a patient has lung cancer (C) based on symptoms like coughing (S) and smoking history (H).

A Bayesian network represents this as:

- $P(C \mid H) \rightarrow$ Probability of having cancer given smoking history.
- $P(S \mid C) \rightarrow$ Probability of coughing given lung cancer.
- $P(H) \rightarrow$ Prior probability of a person being a smoker.

Using Bayes' Theorem, we update our belief when new evidence (e.g., symptoms) is observed.

Bayes' Theorem Formula:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

where:

- $P(H|E)$ = Probability of hypothesis H given evidence E (**posterior probability**).
- $P(E|H)$ = Probability of evidence given hypothesis (**likelihood**).
- $P(H)$ = Prior probability of the hypothesis.
- $P(E)$ = Probability of the evidence.

Use: Helps in reasoning under uncertainty and updating beliefs dynamically.

B. Markov Models & Hidden Markov Models (HMMs)

- Markov Models assume that the future state depends only on the current state, not the past history (Markov Property).
- Hidden Markov Models (HMMs) are used when states are not directly observable (e.g., speech recognition).

Example: Speech Recognition

- The actual spoken word is the hidden state.
- The observed audio signal is the evidence.
- HMMs predict the most likely word sequence based on probabilities.

Use: Useful for time-series data and sequential decision-making.

C. Fuzzy Logic

- Deals with **degrees of truth** rather than strict true/false logic.
- Used in **control systems, robotics, and expert systems**.

Example: Air Conditioner System

- Instead of "hot" or "cold", fuzzy logic defines temperature in **degrees of membership** (e.g., **warm = 0.6, hot = 0.8**).
- Allows for **smooth transitions** in decision-making.

Use: Handles imprecise and vague information well.

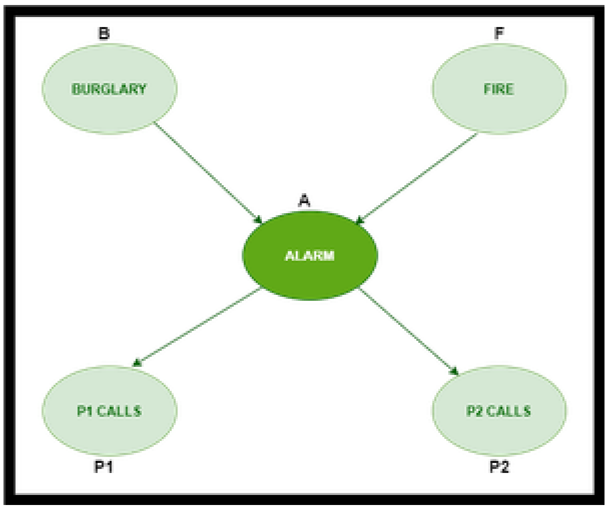
Bayesian Belief Network (BBN) is a graphical model that represents the probabilistic relationships among variables. It is used to handle uncertainty and make predictions or decisions based on probabilities.

- **Graphical Representation**: Variables are represented as nodes in a directed acyclic graph (DAG), and their dependencies are shown as edges.
- **Conditional Probabilities**: Each node's probability depends on its parent nodes, expressed as P(Variable | Parent)*P*(Variable | Parent).
- **Probabilistic Model**: Built from probability distributions, BBNs apply probability theory for tasks like prediction and anomaly detection.

Bayesian Belief Networks are valuable tools for understanding and solving problems involving uncertain events. They are also known as **Bayes networks, belief networks, decision networks, or Bayesian models**.

(Note: A classifier assigns data in a collection to desired categories.)

- Consider this example:



- In the above figure, we have an alarm 'A' – a node, say installed in a house of a person 'gfg', which rings upon two probabilities i.e burglary 'B' and fire 'F', which are – parent nodes of the alarm node. The alarm is the parent node of two probabilities P1 calls 'P1' & P2 calls 'P2' person nodes.
- Upon the instance of burglary and fire, 'P1' and 'P2' call person 'abc', respectively. But, there are few drawbacks in this case, as sometimes 'P1' may forget to call the person 'abc, even after hearing the alarm, as he has a tendency to forget things, quick. Similarly, 'P2', sometimes fails to call the person 'abc, as he is only able to hear the alarm, from a certain distance.

Calculating Conditional Probability of Events in a Bayesian Network

Find the probability that ‘P1’ is true (P1 has called ‘abc), ‘P2’ is true (P2 has called ‘abc) when the alarm ‘A’ rang, but no burglary ‘B’ and fire ‘F’ has occurred.

=> **P (P1, P2, A, ~B, ~F)** [where- P1, P2 & A are ‘true’ events and ‘~B’ & ‘~F’ are ‘false’ events]

[**Note:** The values mentioned below are neither calculated nor computed. They have observed values]

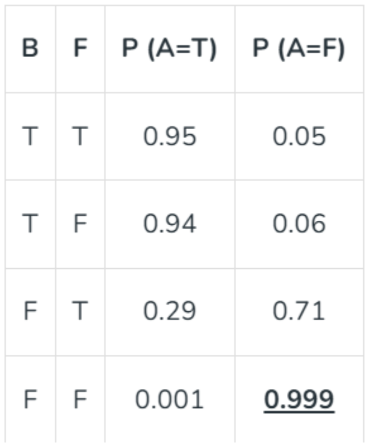
Burglary ‘B’ –

- **P (B=T) = 0.001** (‘B’ is true i.e burglary has occurred)
- **P (B=F) = 0.999** (‘B’ is false i.e burglary has not occurred)

Fire ‘F’ –

- **P (F=T) = 0.002** (‘F’ is true i.e fire has occurred)
- **P (F=F) = 0.998** (‘F’ is false i.e fire has not occurred)

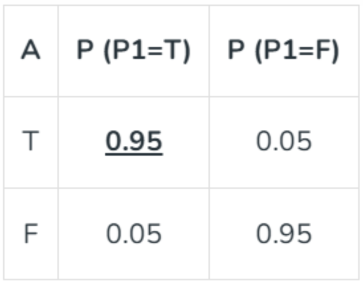
Alarm ‘A’ –



B	F	P (A=T)	P (A=F)
T	T	0.95	0.05
T	F	0.94	0.06
F	T	0.29	0.71
F	F	0.001	**0.999**

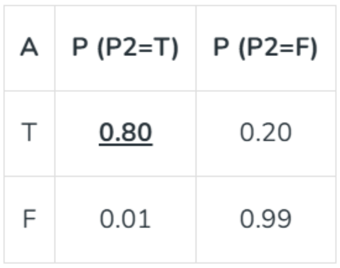
- The alarm ‘A’ node can be ‘true’ or ‘false’ (i.e may have rung or may not have rung). It has two parent nodes burglary ‘B’ and fire ‘F’ which can be ‘true’ or ‘false’ (i.e may have occurred or may not have occurred) depending upon different conditions.
- **Person ‘P1’ –**

The person ‘P1’ node can be ‘true’ or ‘false’ (i.e may have called the person ‘abc or not) . It has a parent node, the alarm ‘A’, which can be ‘true’ or ‘false’ (i.e may have rung or may not have rung ,upon burglary ‘B’ or fire ‘F’).



A	P (P1=T)	P (P1=F)
T	**0.95**	0.05
F	0.05	0.95

- **Person ‘P2’ –**
  The person ‘P2’ node can be ‘true’ or false’ (i.e may have called the person ‘gfg’ or not). It has a parent node, the alarm ‘A’, which can be ‘true’ or ‘false’ (i.e may have rung or may not have rung, upon burglary ‘B’ or fire ‘F’).



A	P (P2=T)	P (P2=F)
T	**0.80**	0.20
F	0.01	0.99

Solution: Considering the observed probabilistic scan –

With respect to the question — **P (P1, P2, A, ~B, ~F)** , we need to get the probability of ‘P1’. We find it with regard to its parent node – alarm ‘A’. To get the probability of ‘P2’, we find it with regard to its parent node — alarm ‘A’.

We find the probability of alarm ‘A’ node with regard to ‘~B’ & ‘~F’ since burglary ‘B’ and fire ‘F’ are parent nodes of alarm ‘A’.

From the observed probabilistic scan, we can deduce –

P (P1, P2, A, ~B, ~F)

= P (P1/A) * P (P2/A) * P (A/~B~F) * P (~B) * P (~F)

= 0.95 * 0.80 * 0.001 * 0.999 * 0.998

= 0.00075

Hidden Markov Model in Machine learning

When working with sequences of data, we often face situations **where we can't directly see the important factors that influence the datasets**. **Hidden Markov Models (HMM)** help solve this problem by predicting these hidden factors based on the observable data

Hidden Markov Model in Machine Learning

It is an statistical model that is used to describe the **probabilistic relationship between a sequence of observations and a sequence of hidden states**. Ike it is often used in situations where the underlying system or process that generates the observations is unknown or hidden, hence it has the name "**Hidden Markov Model**."

An HMM consists of two types of variables: hidden states and observations.

- The **hidden states** are the underlying variables that generate the observed data, but they are not directly observable.
- The **observations** are the variables that are measured and observed.

The relationship between the hidden states and the observations is modeled using a probability distribution. The Hidden Markov Model (HMM) is the relationship between the hidden states and the observations using two sets of probabilities: the transition probabilities and the emission probabilities.

- The **transition probabilities** describe the probability of transitioning from one hidden state to another.
- The **emission probabilities** describe the probability of observing an output given a hidden state.

Hidden Markov Model Algorithm

The Hidden Markov Model (HMM) algorithm can be implemented using the following steps:

- **Step 1: Define the state space and observation space:** The state space is the set of all possible hidden states, and the observation space is the set of all possible observations.
- **Step 2: Define the initial state distribution:** This is the probability distribution over the initial state.
- **Step 3: Define the state transition probabilities:** These are the probabilities of transitioning from one state to another. This forms the transition matrix, which describes the probability of moving from one state to another.
- **Step 4: Define the observation likelihoods:** These are the probabilities of generating each observation from each state. This forms the emission matrix, which describes the probability of generating each observation from each state.

- **Step 5: Train the model:** The parameters of the state transition probabilities and the observation likelihoods are estimated using the Baum-Welch algorithm, or the forward-backward algorithm. This is done by iteratively updating the parameters until convergence.
- **Step 6: Decode the most likely sequence of hidden states:** Given the observed data, the Viterbi algorithm is used to compute the most likely sequence of hidden states. This can be used to predict future observations, classify sequences, or detect patterns in sequential data.
- **Step 7: Evaluate the model:** The performance of the HMM can be evaluated using various metrics, such as accuracy, precision, recall, or F1 score.

To summarise, the HMM algorithm involves defining the state space, observation space, and the parameters of the state transition probabilities and observation likelihoods, training the model using the Baum-Welch algorithm or the forward-backward algorithm, decoding the most likely sequence of hidden states using the Viterbi algorithm, and evaluating the performance of the model.

Hidden Markov Models Explained with a Real Life Example

Hidden Markov Models are close relatives of Markov Chains, but their hidden states make them a unique tool to use when you're interested in determining the probability of a sequence of random variables.

The real world is full of phenomena for which we can see the final outcome, but can't actually observe the underlying factors that generated those outcomes. One example is predicting the weather, determining if it's going to be rainy or sunny tomorrow, based on past weather observations and the observed probabilities of the different weather outcomes.

Although driven by factors we can't observe, with an **Hidden Markov Model** it's possible to model these phenomena as probabilistic systems.

Markov Models with Hidden States

Hidden Markov Models, known as HMM for short, are statistical models that work as a sequence of labeling problems. These are the types of problems that describe the evolution of observable events, which themselves, are dependent on internal factors that can't be directly observed — they are **hidden**[3].

An Hidden Markov Model is made of two distinct stochastic processes, meaning those are processes that can be defined as sequences of random variables — variables that depend on random events.

There’s an **invisible process** and an **observable process**.

The **invisible process** is a Markov Chain, like chaining together multiple **hidden states** that are traversed over time in order to reach an outcome. This is a probabilistic process because all the parameters of the Markov Chain, as well as the score of each sequence, are in fact probabilities[4].

Hidden Markov Models describe the evolution of observable events, which themselves, are dependent on internal factors that can’t be directly observed — they are **hidden**[3]

Just like any other Markov Chain, in order to know which state you’re going next, the only thing that matters is where you are now — in which state of the Markov Chain you’re currently in. None of the previous *history* of states you’ve been in the past matters to understand where you’re going next.

This kind of *short-term* memory is one of the key characteristics of HMMs and it’s called the **Markov Assumption**, indicating that the probability of reaching the next state is only dependent on the probability of the current state.

$$\mathrm{P}(q_i = a \mid \underbrace{q_1 \ldots q_{i-1}}_{\text{history of states traversed}}) = \mathrm{P}(\overbrace{q_i = a}^{\text{next state}} \mid \underset{\text{current state}}{q_{i-1}})$$

Markov Assumption.

The other key characteristic of an HMM, is that it also assumes that each observation is only dependent on the state that produced it therefore, being completely independent from any other state in the chain[5].

*The **Markov Assumption** states that the probability of reaching the next state is only dependent on the probability of the current state.*

This is all great background information on HMM but, what classes of problems are they actually used in?

HMMs help model the behavior of phenomena. Besides modeling and allowing to run simulations, you can also ask different types of questions those phenomena:

- **Likelihood** or **Scoring**, as in, determining the probability of observing a sequence
- **Decoding** the best sequence of states that generated a specific observation
- **Learning** the parameters of the HMM that led to observing a given sequence, that traversed a specific set of states.