UNIT 3

Probabilistic reasoning:

Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge. In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty.

We use probability in probabilistic reasoning because it provides a way to handle the uncertainty that is the result of someone's laziness and ignorance.

In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as "It will rain today," "behavior of someone for some situations," "A match between two teams or two players." These are probable sentences for which we can assume that it will happen but not sure about it, so here we use probabilistic reasoning.

Need for Probabilistic Reasoning in AI

Probabilistic reasoning with artificial intelligence is important to different tasks such as:

- **When there are unpredictable outcomes.**
- **When specifications or possibilities of predicates becomes too large to handle.**
- **When an unknown error occurs during an experiment.**

In probabilistic reasoning, there are two ways to solve problems with uncertain knowledge:

- **Bayes' rule**
- **Bayesian Statistics**

Key Concepts in Probabilistic Reasoning

1. Bayesian Networks

- Imagine a kind of spider web cluttered with factors—one might say, a type of detective board associating suspects, motives, and evidence. This, in a nutshell, is your basic intuition behind a Bayesian network: a graphical model showing the relationships between variables and their conditional probabilities.
- **Advantages:** Bayesian Networks are very effective to express cause and effect and reasoning about missing information. They have found wide applications in medical

diagnosis where symptoms are considered variables which have different grades of association with diseases considered other variables.

2. Markov Models

- Consider a weather forecast. A Markov model predicts the future state of a system from its current state and its past history. For instance, according to a simple Markov model of weather, the probability that a sunny day will be followed by another sunny day is greater than the probability that a sunny day will be followed by a rainy day.
- **Advantages:** Markov models are effective and easy to implement. They are widely used, such as in speech recognition, and they can also be used for prediction, depending on the choice of the previous words, as in the probability of the next word.

3. Hidden Markov Models (HMMs)

- Consider, for example, a weather-predicting scenario that includes states of some kind and yet also includes invisible states, such as humidity. HMMs are a generalization of Markov models in which states are hidden.
- **Advantages:** HMMs are found to be very powerful in cases where hidden variables are taken into account. Such tasks usually involve stock market prediction, where the factors that govern prices are not fully transparent.

4. Probabilistic Graphical Models

- Probabilistic Graphical Models give a broader framework encompassing both Bayesian networks and HMMs. In general, PGMs are an approach for representation and reasoning in a framework of uncertain information, given in graphical structure.
- **Advantages:** PGMs offer a powerful, flexible, and expressive language for doing probabilistic reasoning, which is well suited for complex relationships that may capture many different types of uncertainty.

Probability: Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.

1. $0 \leq P(A) \leq 1$, where P(A) is the probability of an event A.
1. $P(A) = 0$, indicates total uncertainty in an event A.

1. P(A) =1, indicates total certainty in an event A.

   We can find the probability of an uncertain event by using the below formula.

$$\textbf{Probability of occurrence} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$$

- P(~A) = probability of a not happening event.
- P(~A) + P(A) = 1.

Event: Each possible outcome of a variable is called an event.

Sample space: The collection of all possible events is called sample space.

Random variables: Random variables are used to represent the events and objects in the real world.

Prior probability: The prior probability of an event is probability computed before observing new information.

Posterior Probability: The probability that is calculated after all evidence or information has taken into account. It is a combination of prior probability and new information.

Conditional probability

Conditional probability is the probability that depends on a previous result or event. Due to this fact, they help us understand how events are related to each other. Simply put, conditional probability tells us the likelihood of the occurrence of an event based on the occurrence of some previous outcome.

Conditional probability is a probability of occurring an event when another event has already happened.

Conditional Probability Formula

two events A and B, then the **formula for conditional probability** of A when B has already occurred is given by:

$$P(A|B) = P(A \cap B) / P(B)$$

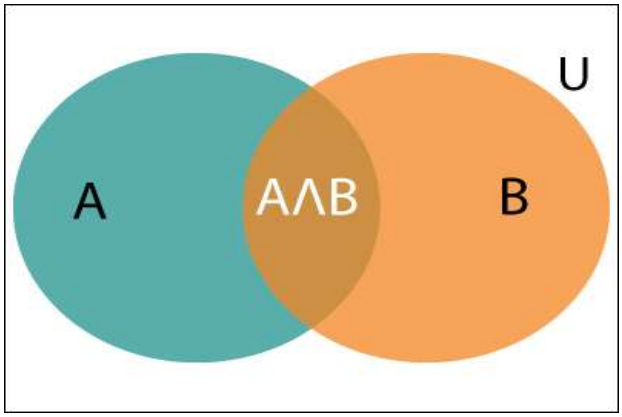
Where,

- ***P (A ∩ B)*** *represents the probability of both events A and B occurring simultaneously.*

- ***P(B)*** *represents the probability of event B occurring.*
- If the probability of A is given and we need to find the probability of B, then it will be given as:

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

- It can be explained by using the below Venn diagram, where B is occurred event, so sample space will be reduced to set B, and now we can only calculate event A when event B is already occurred by dividing the probability of **P(A∧*B*) by P(B)**



Example: consider the case of rolling two dice, sample space of this event is as follows:

{(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6),
(2, 1), (2, 2), (2, 3), (2, 4), (2, 5), (2, 6),
(3, 1), (3, 2), (3, 3), (3, 4), (3, 5), (3, 6),
(4, 1), (4, 2), (4, 3), (4, 4), (4, 5), (4, 6),
(5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 6),
(6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 6)}

Now, consider an event A = getting 3 on the first die and B = getting a sum of 9.

Then the probability of getting 9 when on the first die it's already 3 is P(B | A),

which can be calculated as follows:

All the cases for the first die as 3 are (3, 1), (3, 2), (3, 3), (3, 4), (3, 5), (3, 6).

In all of these cases, only one case has a sum of 9.

Thus, P (B | A) = 1/36.

In case, we have to find P (A | B),

All cases where the sum is 9 are (3, 6), (4, 5), (5, 4), and (6, 3).

In all of these cases, only one case has 3 on the first die i.e., (3, 6)

Thus, P(A | B) = 1/36.

Bayesian Network

Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. We can define a Bayesian network as:

"A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph."

It is also called a **Bayes network, belief network, decision network**, or **Bayesian model**.

Bayesian networks are probabilistic, because these networks are built from a **probability distribution**, and also use probability theory for prediction and anomaly detection.

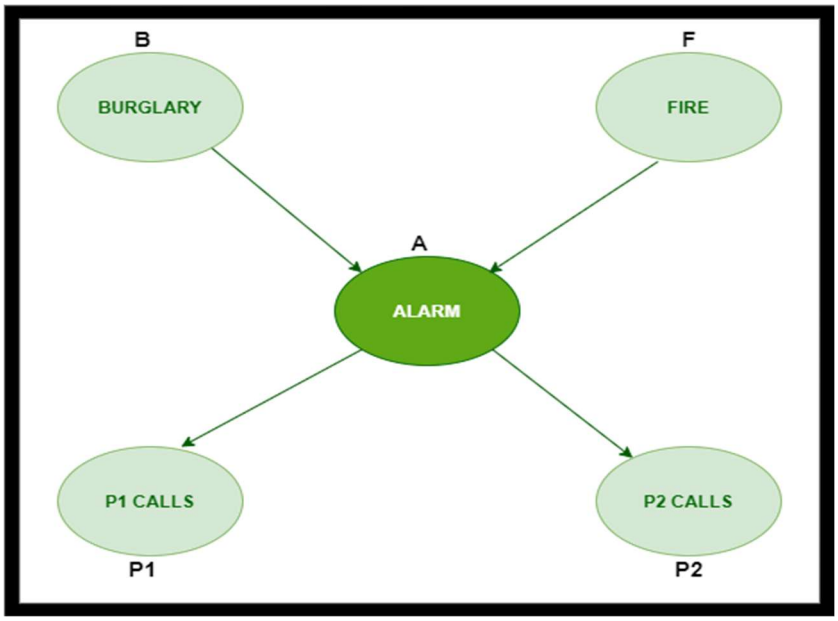
Bayesian Network can be used for building models from data and experts opinions, and it consists of two parts:

- **Directed Acyclic Graph**
- **Table of conditional probabilities.**

The Bayesian network has mainly two components:

- **Causal Component**
- **Actual numbers**

Consider this example:



Q) Find the probability that 'P1' is true (P1 has called 'gfg'), 'P2' is true (P2 has called 'gfg') 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' –

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

- The person 'P1' 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').

Person 'P2' –

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

- 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').

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

AI Inference

AI inference involves applying a trained machine learning model to make predictions or decisions based on new, unseen data. This phase contrasts with the training period, where a model learns from a dataset by adjusting its parameters (weights and biases) to minimize errors, preparing it for real-world applications.

In artificial intelligence, we need intelligent computers which can create new logic from old logic or by evidence, so generating the conclusions from evidence and facts is termed as Inference.

Inference Rules and Terminologies

In AI, inference rules serve as guiding principles for deriving valid conclusions from existing data. These rules underpin the construction of proofs, which constitute chains of reasoning leading to desired outcomes. Within these rules lie key terminologies that delineate relationships between propositions connected by various logical connectives:

- **Implication**: Symbolized by A → B, implication denotes that proposition A implies proposition B, suggesting a cause-and-effect relationship.
- **Converse**: Flipping the implication, placing B on the left and A on the right (B → A), though the converse doesn't ensure the original implication's validity.
- **Contrapositive**: The negation of the converse (¬B → ¬A), offering an equivalent implication with both propositions negated.

- **Inverse**: Symbolized by ¬A → ¬B, the inverse represents the negation of the original implication, albeit not guaranteeing its truth.

Types of Inference Rules

1. **Modus Ponens**: This rule dictates that if "A implies B" and "A" is true, then "B" must also be true, exemplifying a crucial rule of inference.
2. **Modus Tollens**: Stating that if "A implies B" and "B" is false, then "A" must be false, illustrating the negation of the consequent.
3. **Hypothetical Syllogism**: Involving reasoning from one conditional statement to another, this rule leverages the first statement to infer conclusions about the second, showcasing a chain of logical deductions.
4. **Disjunctive Syllogism**: Dealing with "or" statements, this method infers the truth of one proposition by negating the other, revealing a logical disjunction.
5. **Constructive Dilemma**: Entailing two conditional statements and a statement about their alternatives, this rule enables the inference of logical conclusions based on potential scenarios.
6. **Destructive Dilemma**: Addressing "if-then" statements and their negations, this method identifies flaws by showcasing that if an outcome isn't true, then one of the initial assumptions must be flawed.

Applications of Inference in AI

1. **Medical Research and Diagnoses**: AI aids in medical research and diagnoses by analyzing patient data to provide optimized treatment plans and prognoses.
2. **Recommendation Systems and Personalized Advertisements**: E-commerce platforms utilize inference to suggest products based on user preferences, enhancing user experience and engagement.
3. **Self-Driving Vehicles**: Inference enables self-driving cars to interpret sensor data and navigate through dynamic environments safely and efficiently.

Temporal Models

Temporal models are used to represent probabilistic relationships between sequences of random variables that change over time. These models capture the dynamics and

dependencies of data points within a sequence, allowing for the prediction and analysis of future states based on past and present observations.

Key Components of Temporal Models:

- **States**: These represent the possible conditions of the system at different times.
- **Observations**: These are the data points that are directly measured or perceived.
- **Transitions**: These are the probabilities from one state to another over time.
- **Emissions**: These are the probabilities of observing certain data given the system's state.

Hidden Markov Model

Hidden Markov Models (HMMs) are a type of probabilistic model that are commonly used in machine learning for tasks such as speech recognition, natural language processing, and bioinformatics. They are a popular choice for modelling sequences of data because they can effectively capture the underlying structure of the data, even when the data is noisy or incomplete.

A Hidden Markov Model (HMM) is a probabilistic model that consists of a sequence of hidden states, each of which generates an observation. The hidden states are usually not directly observable, and the goal of HMM is to estimate the sequence of hidden states based on a sequence of observations. An HMM is defined by the following components:

- A set of N hidden states, S = {s1, s2, ..., sN}.
- A set of M observations, O = {o1, o2, ..., oM}.
- An initial state probability distribution, ? = {?1, ?2, ..., ?N}, which specifies the probability of starting in each hidden state.
- A transition probability matrix, A = [aij], defines the probability of moving from one hidden state to another.
- An emission probability matrix, B = [bjk], defines the probability of emitting an observation from a given hidden state.

  The basic idea behind an HMM is that the hidden states generate the observations, and the observed data is used to estimate the hidden state sequence. This is often referred to as the **forward-backwards algorithm.**

Applications of HMM

- **Speech Recognition**

  One of the most well-known applications of HMMs is speech recognition. In this field, HMMs are used to model the different sounds and phones that makeup speech. The hidden states, in this case, correspond to the different sounds or

phones, and the observations are the acoustic signals that are generated by the speech. The goal is to estimate the hidden state sequence, which corresponds to the transcription of the speech, based on the observed acoustic signals. HMMs are particularly well-suited for speech recognition because they can effectively capture 00the underlying structure of the speech, even when the data is noisy or incomplete. In speech recognition systems, the HMMs are usually trained on large datasets of speech signals, and the estimated parameters of the HMMs are used to transcribe speech in real time.

- **Natural Language Processing**
  Another important application of HMMs is natural language processing. In this field, HMMs are used for tasks such as **part-of-speech tagging, named entity recognition, and text classification**. In these applications, the hidden states are typically associated with the underlying grammar or structure of the text, while the observations are the words in the text. The goal is to estimate the hidden state sequence, which corresponds to the structure or meaning of the text, based on the observed words. HMMs are useful in natural language processing because they can effectively capture the underlying structure of the text, even when the data is noisy or ambiguous. In natural language processing systems, the HMMs are usually trained on large datasets of text, and the estimated parameters of the HMMs are used to perform various NLP tasks, such as text classification, part-of-speech tagging, and named entity recognition.
- **Bioinformatics**
  HMMs are also widely used in bioinformatics, where they are used to model sequences of DNA, RNA, and proteins. The hidden states, in this case, correspond to the different types of residues, while the observations are the sequences of residues. The goal is to estimate the hidden state sequence, which corresponds to the underlying structure of the molecule, based on the observed sequences of residues. HMMs are useful in bioinformatics because they can effectively capture the underlying structure of the molecule, even when the data is noisy or incomplete. In bioinformatics systems, the HMMs are usually trained on large datasets of molecular sequences, and the estimated parameters of the HMMs are used to predict the structure or function of new molecular sequences.
- **Finance**
  Finally, HMMs have also been used in finance, where they are used to model stock prices, interest rates, and currency exchange rates. In these applications, the hidden states correspond to different economic states, such as bull and bear markets, while the observations are the stock prices, interest rates, or exchange rates. The goal is to estimate the hidden state sequence, which corresponds to the

underlying economic state, based on the observed prices, rates, or exchange rates. HMMs are useful in finance because they can effectively capture the underlying economic state, even when the data is noisy or incomplete. In finance systems, the HMMs are usually trained on large datasets of financial data, and the estimated parameters of the HMMs are used to make predictions about future market trends or to develop investment strategies.