Unit-5

Uncertain knowledge and Learning Uncertainty: Acting under Uncertainty, Basic Probability Notation, Inference Using Full Joint Distributions, Independence,

Bayes' Rule and its Use. Probabilistic Reasoning: Representing Knowledge in an Uncertain Domain, The Semantics of Bayesian Networks, Efficient Representation of Conditional Distributions, Approximate Inference in Bayesian Networks, Relational and First-Order Probability,

Other Approaches to Uncertain Reasoning; Dempster-Shafer theory

Causes of uncertainty:

Following are some leading causes of uncertainty to occur in the real world.

- Information occurred from unreliable sources.
- Experimental Errors
- Equipment fault
- Temperature variation
- Climate change.

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.

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.**

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

We can find the probability of an uncertain event by using the below formula.
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 a probability of occurring an event when another event has already happened

For example : To calculate the event A when event B has already occurred, "the probability of A under the conditions of B", it can be written as:
Where **P(A⋀B)= Joint** probability of A and B
P(B)= Marginal probability of B.

Example:
In a class, there are 70% of the students who like English and 40% of the students who likes English and mathematics, and then what is the percent of students those who like English also like mathematics?
Solution:
Let, A is an event that a student likes Mathematics
B is an event that a student likes English.
Hence, 57% are the students who like English also like Mathematics.

Some of the reasons for reasoning under uncertainty:

- **True uncertainty**. E.g., flipping a coin.
- **Theoretical ignorance**. There is no complete theory which is known about the problem domain. E.g., medical diagnosis.
- **Laziness**. The space of relevant factors is very large, and would require too much work to list the complete set of antecedents and consequents. Furthermore, it would be too hard to use the enormous rules that resulted.
- **Practical ignorance**. Uncertain about a particular individual in the domain because all of the information necessary for that individual has not been collected.

Bayes' theorem:

Bayes' theorem is also known as **Bayes' rule, Bayes' law**, or **Bayesian reasoning**, which determines the probability of an event with uncertain knowledge.

In probability theory, it relates the conditional probability and marginal probabilities of two **random events**.

Bayes' theorem was named after the British mathematician **Thomas Bayes**. The **Bayesian inference** is an application of Bayes' theorem, which is fundamental to Bayesian statistics **It is a way to calculate the value of P(B|A) with the knowledge of P(A|B).**

Bayes' theorem allows **updating the probability prediction** of an event by observing **new information of the real world**.

Example: If cancer corresponds to one's age then by using Bayes' theorem, we can determine the probability of cancer more accurately with the help of age.

Bayes' theorem can be derived **using product rule and conditional probability** of event A with known event B:

As from product rule we can write:

1. P(A ∧ B)= P(A|B) P(B) or

Similarly, the probability of event B with known event A:

1. P(A ∧ B)= P(B|A) P(A)

Equating right hand side of both the equations, we will get:

$$\boxed{P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}} \quad \text{....(a)}$$

The above equation (a) is called as **Bayes' rule** or **Bayes' theorem**. This equation is basic of most modern AI systems for **probabilistic inference**.

It shows the simple relationship **between joint and conditional probabilities**. Here,
P(A|B) is known as **posterior**, which we need to calculate, and it will be read as Probability of hypothesis **A** when we have occurred an evidence **B**.
P(B|A) is called the likelihood, in which we consider that hypothesis is true, then we calculate the probability of evidence.
P(A) is called the **prior probability**, probability of hypothesis before considering the evidence P(B) is called **marginal probability**, pure probability of an evidence.

In the equation (a), in general, we can write P (B) = P(A)*P(B|Ai), hence the Bayes' rule can be written as:

$$P(A_i|B) = \frac{P(A_i)*P(B|A_i)}{\sum_{i=1}^{k} P(A_i)*P(B|A_i)}$$

Applying Bayes' rule:

Bayes' rule allows us to compute the single term P(B|A) in terms of P(A|B), P(*B*), and P(A). This is very useful in cases where we have a good probability of these three terms and want to determine the fourth one.

Suppose we want to perceive the effect of some unknown cause, and want to compute that cause, then the Bayes' rule becomes:

$$P(\text{cause}|\text{effect}) = \frac{P(\text{effect}|\text{cause})\ P(\text{cause})}{P(\text{effect})}$$

Example-1:

Question: what is the probability that a patient has diseases meningitis with a stiff neck?

Given Data:

A doctor is aware that disease meningitis causes a patient to have a stiff neck, and it occurs 80% of the time. He is also aware of some more facts, which are given as follows:

- The Known probability that a patient has meningitis disease is 1/30,000.
- The Known probability that a patient has a stiff neck is 2%.

Let a be the proposition that patient has stiff neck and b be the proposition that patient has meningitis. , so we can calculate the following as:

P(a|b) = 0.8

P(b) = 1/30000

P(a)= .02

$$P(b|a) = \frac{P(a|b)P(b)}{P(a)} = \frac{0.8*(\frac{1}{30000})}{0.02} = 0.001333333.$$

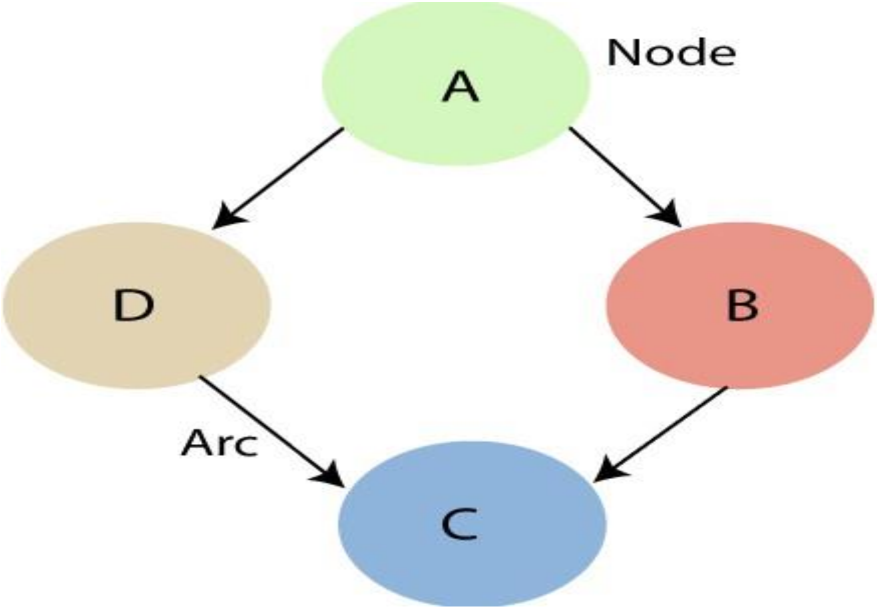
Bayesian Network

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 generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an **Influence diagram**.

A Bayesian network graph is made up of nodes and Arcs (directed links), where:



- Each **node** corresponds to the random variables, and a variable can be **continuous** or **discrete**.
  Arc or directed arrows represent the causal relationship or conditional probabilities between random variables
- **In the above diagram, A, B, C, and D are random variables represented by the nodes of the network graph.**
- **If we are considering node B, which is connected with node A by a directed arrow,**

then node A is called the parent of Node B.

- **Node C is independent of node A.**

The Bayesian network has mainly **two components**:

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

Each node in the Bayesian network has condition probability distribution **$P(X_i | Parent(X_i))$**, which determines the effect of the parent on that node.

Representing Belief about Propositions

- Rather than reasoning about the truth or falsity of a proposition, reason about the belief that **a proposition or event is true or false**
- For each primitive proposition or event, attach a **degree of belief** to the sentence
- Use **probability theory** as a formal means of manipulating degrees of belief
- Given a proposition A, assign a probability, P(A), such that 0 <= P(A) <= 1, where if A is true, P(A)=1, and if A is false, P(A)=0. Proposition A must be either true or false, but P(A) summarizes our degree of belief in A being true/false.

Example

- P(Weather=Sunny) = 0.7 means that we believe that the weather will be Sunny with 70% certainty. In this case Weather is a random variable that can take on values in a domain such as {Sunny, Rainy, Snowy, Cloudy}.

- P(Cavity=True) = 0.05 means that we believe there is a 5% chance that a person has a cavity. Cavity is a Boolean random variable since it can take on possible values *True* and *False*.

Example

P(A=a ^ B=b) =P(A=a, B=b) = 0.2, where A=My_Mood, a=happy, B=Weather, and b=rainy, means that there is a 20% chance that when it's raining my mood is happy.

- We will assume that in a given problem domain, the programmer and expert identify all of the relevant propositional variables that are needed to reason about the domain.
- Each of these will be represented as a **random variable**, i.e., a variable that can take on values from a set of mutually exclusive and exhaustive values called the **sample space** or **partition** of the random variable. Usually this will mean a sample space {*True*, *False*}.
- **For example**, the proposition *Cavity* has possible values *True* and *False* indicating whether a given patient has a cavity or not. **A random variable** that has True and False as its possible values is called a **Boolean random variable**.

Acting under Uncertainty

Acting under uncertainty is one of the most significant challenges and active areas of research in Artificial Intelligence.
Real-world environments are inherently unpredictable, with incomplete information, noisy sensors, and dynamic changes
AI agents cannot simply rely on deterministic rules or perfect knowledge.

- **Probability Theory:** To quantify uncertainty.
- **Decision Theory:** To make optimal choices under uncertainty.
- **Utility Theory:** To define the agent's preferences for different outcomes.

Why Acting Under Uncertainty is Crucial in AI

- **Incomplete Information:** Agents rarely have full knowledge of the state of the world (e.g., a robot might not know the exact location of all obstacles).
- **Noisy Sensors:** Sensor readings are often imperfect and can provide erroneous data (e.g., a vision system misidentifying an object).
- **Stochastic Environments:** The world itself can be unpredictable, with events occurring probabilistically (e.g., a dice roll, traffic patterns).
- **Unknown Outcomes of Actions:** Actions do not always have deterministic effects (e.g., pressing a button might sometimes fail).
- **Hidden States:** Important variables might be unobservable (e.g., the true health of a patient).
- **Opponent Actions:** In multi-agent systems, other agents' actions are often uncertain.

Components for Acting Under Uncertainty :

1. Belief State
2.Actions
3.Utility Function

1. **Belief State (Beliefs about the World):**
   - The agent's current knowledge about the state of the world, represented as a probability distribution over possible states.
   - This is often maintained using probabilistic inference (e.g., updating beliefs in a Bayesian Network based on new evidence).
   - For dynamic systems, this involves tracking the state over time (e.g., using a Kalman Filter or Particle Filter in robotics).
2. **Actions:**

    - A set of possible actions the agent can perform.
    - Each action has **stochastic outcomes** (i.e., its effects are not perfectly predictable).
    - An **action model** describes $P(Result|Action,Current\ State)$. This quantifies the probability of transitioning to a new state given an action and the current state.

3. **Utility Function (Preferences):**
    - Quantifies the desirability of different states or outcomes for the agent.
    - Typically a numerical value, where higher values indicate more preferred outcomes.
    - Example: In a medical diagnosis system, a correct diagnosis leading to recovery has high utility, while a wrong diagnosis causing harm has low utility (or high negative utility).

Decision Making Under Uncertainty (Decision Theory) :

Maximizes the **Expected Utility**.

- **Expected Utility (EU) of an Action:** The sum of the utilities of all possible outcomes of an action, weighted by their probabilities.

  $EU(A)=s'\sum P(Result=s'|Action=A,Current\ Beliefs)\times U(s')$

  where s' is a possible next state, and $U(s')$ is the utility of that state.

- **Optimal Decision:** The agent chooses the action $A\ast$ such that $A\ast=argmaxAEU(A)$.

Several AI formalisms are designed to enable acting under uncertainty:

1. **Decision Networks**
2. **Markov Decision Processes (MDPs)**
3. **Partially Observable Markov Decision Processes (POMDPs)**
4. **Reinforcement Learning (RL)**
5. **Game Theory**

Challenges in Acting Under Uncertainty :

- **Computational Complexity:** Calculating expected utilities can be computationally intensive, especially for large state spaces or long planning horizons (e.g., POMDPs are PSPACE-complete).
- **Modeling Difficulty:** Accurately defining probability distributions (transition models, sensor models) and utility functions can be challenging.

- **Exploration vs. Exploitation:** In learning settings (RL), the agent faces a dilemma: should it explore new actions to gain more information about the environment, or exploit its current knowledge to maximize immediate reward?
- **Dynamic Environments:** Adapting to continuously changing probabilities or utility functions.

Basic Probability Notation

Probability provides the mathematical framework for handling uncertainty in AI. Understanding its notation is fundamental for working with probabilistic models like Bayesian Networks, Markov Models, and for understanding concepts in machine learning.

1. Events and Variables

- **Events:** Specific outcomes or propositions.
    - Notation: Capital letters (e.g., `A`, `B`), or descriptive names (e.g., `Rain`, `HasCavity`).
    - Example: `A` = "It will rain tomorrow". `HasCavity` = "The patient has a cavity".
- **Random Variables:** Quantities whose values are subject to variations due to chance. They can be discrete (e.g., `CoinFlip` can be `Heads` or `Tails`) or continuous (e.g., `Temperature`).
    - Notation: Capital letters (e.g., `X`, `Y`, `Z`), or descriptive names (e.g., `Weather`, `Symptom`).
    - Values of Variables: Denoted by lowercase letters (e.g., `x`, `y`) or specific constant values.
        - Example: `Weather = sunny`, `Symptom = fever`.

2. Probability of an Event/Value

- **Notation:** `P(A)`
- **Meaning:** The probability that event `A` occurs.
- **For Variables:** `P(Weather = rainy)` is the probability that the `Weather` variable takes the value `rainy`.
- **Range:** All probabilities are between 0 and 1, inclusive: $0 \le P(A) \le 1$.
    - `P(A) = 0`: Event `A` is impossible.
    - `P(A) = 1`: Event `A` is certain.

3. **Joint Probability**

- **Notation:** `P(A, B)` or `P(A ∩ B)`
- **Meaning:** The probability that *both* event `A` and event `B` occur simultaneously.
- **For Variables:** `P(Cavity = true, Toothache = true)` is the probability that a patient has both a cavity and a toothache.
- **Full Joint Distribution:** `P(X_1, X_2, ..., X_n)` represents the probability of a specific assignment of values to all variables in the set. This table theoretically defines all probabilistic relationships.

4.**Conditional Probability**

- **Notation:** `P(A | B)`
- **Meaning:** The probability of event `A` occurring *given that* event `B` has already occurred or is known to be true. `B` is the "evidence."
- **For Variables:** `P(Cavity = true | Toothache = true)` is the probability of a cavity given a toothache.
- **Formula:** $P(A|B)=P(B)P(A,B)$ (provided $P(B)>0$).
  - This formula is fundamental for inference in AI, allowing systems to update beliefs based on new information.

5. **Marginal Probability**

- **Meaning:** The probability of a single event or variable value, calculated by summing (for discrete variables) or integrating (for continuous variables) over all possible outcomes of other variables in a joint distribution.
- **Formula (Discrete):** $P(A)=\sum bP(A,B=b)$
  - Example: If you have `P(Weather, Temperature)`, then `P(Weather = sunny)` is the sum of `P(Weather = sunny, Temperature = T)` for all possible values of `T`.

6.**Probability Distribution**

- **Notation:** `P(X)`
- **Meaning:** Represents the entire set of probabilities for all possible values that a random variable `X` can take.
  - For a discrete variable, this is a **Probability Mass Function (PMF)**, often represented as a table.
    - Example: `P(Weather)` could be `{sunny: 0.6, cloudy: 0.3, rainy: 0.1}`.
  - For a continuous variable, this is a **Probability Density Function (PDF)**.

7.**Conditional Independence**

- **Notation:** $A\perp B|C$
- **Meaning:** Event `A` is conditionally independent of event `B` given event `C`. This implies that if `C` is known, then knowing `B` provides no additional information about `A`.
- **Mathematical Implication:** $P(A|B,C)=P(A|C)$

Inference Using Full Joint Distributions

Inference using full joint distributions refers to the theoretical foundation for answering any probabilistic query about a set of random variables.

A full joint probability distribution for a set of random variables X1,X2,…,Xn is a table (or function) that specifies the probability of every possible **complete assignment of values** to all of these variables.

- If each variable Xi has di possible values, then the FJPD table will have ∏i=1ndi entries.
- The sum of all probabilities in the FJPD must equal 1.

Example: Consider a domain with three binary random variables:

- `Cavity` (C): True/False
- `Toothache` (T): True/False
- `Catch` (H, for the dentist's probe catching): True/False

A full joint distribution for these three variables would look like this:

Cavity (C)	Toothache (T)	Catch (H)	P(C, T, H)
False	False	False	0.576
False	False	True	0.064
False	True	False	0.008
False	True	True	0.016
True	False	False	0.108
True	False	True	0.012
True	True	False	0.072
True	True	True	0.144
Total			**1.000**

Advantages of Inference Using Full Joint Probability Distributions

- **Completeness:** The FJPD contains all possible information about the probabilistic relationships between variables. Any possible probabilistic query can be answered.
- **Simplicity of Concept:** The underlying mathematical operations (summation and division) are straightforward.

How Inference is Performed Using an FJPD :

1. **Marginalization**
2. **Conditioning**

1. **Marginalization (Summing Out):** To find the probability of a subset of variables, you sum the joint probabilities over all possible values of the other (unqueried, unobserved) variables.
    - **Rule:** $P(Y)=\sum z\in ZP(Y,Z=z)$, where Y is the set of variables you're interested in, and Z is the set of all other variables.

    - **Example:** To find P(Cavity=True) from the table:

      P(C=True)=P(C=T, T=F, H=F)+P(C=T, T=F, H=T)+P(C=T, T=T, H=F)+P(C=T, T=T, H=T) P(C=True)= **0.108+0.012+0.072+0.144=0.336**

2. **Conditioning:** To find a conditional probability P(Query∣Evidence), you first compute the joint probability of the query and the evidence, and then normalize it.
    - **Rule:** $P(Q|E)=P(E)P(Q,E)$
    - Where P(Q,E) can be found by marginalizing the FJPD over all variables *not* in Q or E.
    - And P(E) can be found by marginalizing the FJPD over all variables *not* in E.
    - **Normalization Constant (α):** Often, we calculate P(Q,E) for all possible values of Q, and then divide by a constant (α) to ensure the probabilities for Q sum to 1. This means you don't explicitly need P(E). $P(Q|E)=\alpha\sum\text{Hidden}\ P(Q,E,\text{Hidden})$
    - **Example:** To find P(Cavity=True∣Toothache=True):
        1. **Find P(Cavity=True, Toothache=True):** Sum entries where C=True and T=True. P(C=T, T=T)=P(C=T, T=T, H=F)+P(C=T, T=T, H=T) P(C=T, T=T)=0.072+0.144=0.216
        2. **Find P(Cavity=False, Toothache=True):** Sum entries where C=False and T=True. P(C=F, T=T)=P(C=F, T=T, H=F)+P(C=F, T=T, H=T) P(C=F, T=T)=0.008+0.016=0.024
        3. **Find P(Toothache=True):** This is P(C=T, T=T)+P(C=F, T=T) P(T=T)=0.216+0.024=0.240
        4. **Calculate the conditional probability:** P(C=T∣T=T)=P(T=T)P(C=T, T=T)=0.2400.216=0.9

Independence

In Artificial Intelligence, the concept of **independence** is absolutely fundamental, particularly in the fields of probabilistic reasoning, machine learning, and knowledge representation

It refers to the idea that certain variables, events, or pieces of information do not influence or provide information about each other.

several key facts of independence in AI

1. Statistical Independence (Marginal Independence)

This is the most basic form, derived directly from probability theory.

Two random variables, A and B, are statistically independent if knowing the value of one provides no information about the value of the other. Mathematically:

- $P(A \cap B)=P(A)P(B)$
- Equivalently, $P(A|B)=P(A)$ (if $P(B)>0$)
- And $P(B|A)=P(B)$ (if $P(A)>0$)

Example: If "whether it rains today" and "the outcome of a coin flip" are truly independent, then knowing the coin flip result tells you nothing about the rain.

2. Conditional Independence

This is a more powerful and frequently used concept in AI, especially in graphical models.

Two random variables, A and B, are conditionally independent given a third variable (or set of variables) C, if once the value of C is known, A provides no additional information about B, and vice-versa. Mathematically:

- $P(A \cap B|C)=P(A|C)P(B|C)$
- Equivalently, $P(A|B,C)=P(A|C)$ (if $P(B \cap C)>0$)
- And $P(B|A,C)=P(B|C)$ (if $P(A \cap C)>0$)

- **Notation:** $A \perp B|C$
- Conditional independence allows us to simplify complex joint probability distributions into a product of simpler conditional distributions. This is the cornerstone of Bayesian Networks and Markov Random Fields.
- **Example:**
  - "Toothache" and "Catch" (a dental probe catching on a tooth) are often conditionally independent given "Cavity." If you know whether the patient has a cavity, knowing about their toothache doesn't give you any more information about whether the probe will catch (and vice-versa).
  - However, "Toothache" and "Catch" are *not* marginally independent, as they both indicate a higher chance of "Cavity."

Representing Knowledge in an Uncertain Domain

Representing knowledge in an uncertain domain is a fundamental challenge in AI, as real-world environments are rarely deterministic or fully observable.

AI systems need to reason and make decisions despite incomplete, noisy, or ambiguous information

How knowledge is represented in uncertain domains in AI:

1. Probabilistic Representations

- ***Bayesian Networks***
- ***Markov Random Fields (MRFs) / Undirected Graphical Models***
- **Dynamic Bayesian Networks (DBNs):**
- **Hidden Markov Models (HMMs):**

- ***Bayesian Networks (BNs):***
  - Directed Acyclic Graphs (DAGs) where nodes represent random variables and directed edges represent direct causal or probabilistic dependencies. Each node has a Conditional Probability Distribution (CPD) that quantifies the probability of its values given its parents' values.

    Advantages: Powerful for representing causal relationships, allows for efficient inference (both exact and approximate), can handle missing data, and supports learning from data.

    Example: A medical diagnosis BN where `Fever` depends on `Flu` and `Cold`, and `Flu` depends on `ExposureToFlu`.

- ***Markov Random Fields (MRFs) / Undirected Graphical Models:***

  Undirected graphs where nodes are variables and edges represent statistical dependencies. The relationships are defined by potential functions (factors) over cliques (fully connected subgraphs).

  - **Knowledge Representation:**
    - Symmetrical dependencies. An edge between A and B means they are directly related.
    - Potential functions (or factors) assign "scores" to configurations of variables within cliques, which can be converted to probabilities.
  - **Advantages:** Well-suited for representing relationships that are not necessarily causal or directed (e.g., image pixels, social network connections). Basis for techniques like Conditional Random Fields (CRFs) for sequence labeling.

- **Dynamic Bayesian Networks (DBNs):**

  Extend BNs to model systems that change over time. They typically consist of "time slices" of variables, with dependencies within a slice and between consecutive slices.

  Defines a stationary process (how variables evolve from one time step to the next).

- **Advantages:** Crucial for time-series analysis, speech recognition, robotics (state estimation), and tracking.

- **Hidden Markov Models (HMMs):**
  - A special case of DBNs where there's a sequence of unobserved (hidden) states and a sequence of observed variables that depend on the hidden states.
  - Transition probabilities between hidden states, and emission probabilities from hidden states to observations.
  - **Advantages:** Widely used in speech recognition, bioinformatics (sequence alignment), and natural language processing.

2. Fuzzy Logic and Fuzzy Systems

Instead of binary (true/false) truth values, fuzzy logic allows for degrees of truth, represented by values between 0 and 1. Fuzzy sets define membership functions.

- **Knowledge Representation:**
  - **Fuzzy Rules:** Express knowledge using linguistic terms and fuzzy implications (e.g., "IF temperature IS high AND pressure IS low THEN fan_speed IS medium").
  - **Membership Functions:** Define what "high" or "low" means in terms of a continuous variable.
- **Advantages:** Good for capturing imprecise human knowledge and reasoning with vagueness. Often used in control systems and expert systems where human intuition is paramount.
- **Limitations:** Does not have the strong probabilistic semantics for uncertainty; rather, it handles vagueness. Difficult to learn from data systematically compared to probabilistic methods.

3. Dempster-Shafer Theory (Evidence Theory)

- Provides a framework for reasoning with partial belief and conflicting evidence. It introduces concepts like *belief* (support for a proposition) and *plausibility* (extent to which evidence doesn't contradict a proposition). Uncertainty is represented by an interval [belief, plausibility].
- **Knowledge Representation:** Assigns *mass* (basic probability assignment) to subsets of possible outcomes, not just individual outcomes.
- **Advantages:** Can distinguish between uncertainty due to ignorance (lack of evidence) and uncertainty due to conflict.
- **Limitations:** Combination rules can be complex; can lead to counter-intuitive results in some cases of strong conflict. Less widely adopted than probability theory.

The Semantics of Bayesian Networks

1. Representation of a Joint Probability Distribution

2. Encoding of Conditional Independence Statements

1. Representation of a Joint Probability Distribution

The primary semantic interpretation of a Bayesian Network is that it provides a **compact factorization of the full joint probability distribution** over all the variables in the network.

- **Variables as Nodes:** Each node in the network represents a random variable (e.g., "Has Cavity," "Toothache," "Catch" in a dental diagnosis scenario). These variables can be observable quantities, latent variables, or hypotheses.
- **Directed Acyclic Graph (DAG) as Structure:** The connections between nodes are directed links (arrows), and the graph must be acyclic (no loops). These links signify **direct conditional dependencies**. An arrow from node A to node B means that A directly influences B, or B is directly dependent on A.
- **Conditional Probability Distributions (CPDs) as Parameters:** For each node Xi, there's an associated **Conditional Probability Distribution (CPD)**, P(Xi∣Parents(Xi)). This table or function quantifies the probability of Xi taking on certain values, given the values of its direct parents in the graph.
    - If a node has no parents (a root node), its CPD is simply its prior marginal probability distribution, P(Xi).
- **The Joint Probability Formula:** The power of a Bayesian Network lies in how it defines the full joint probability distribution. The joint probability of all variables X1 ,X2,…,Xn is given by the product of the conditional probability distributions of each node given its parents:

  P(X1,X2,…,Xn)=i=1∏nP(Xi∣Parents(Xi))

  This formula is a direct consequence of the network structure and the conditional independence assumptions it encodes.

2. Encoding of Conditional Independence Statements:

A crucial aspect of Bayesian Network semantics is their ability to represent **conditional independence relationships**.

- **Intuitive Meaning of Arrows:** An arrow from X to Y usually means that X has a direct influence on Y. However, the absence of an arrow between two nodes is just as important.
- **D-separation:** The structure of the DAG (the nodes and links) implicitly specifies a set of conditional independence assertions. The concept of **d-separation** (directional separation) is used to determine whether two sets of variables are conditionally independent given a third set of variables. If two variables X and Y are d-separated by a set of evidence variables Z, then X and Y are conditionally independent given Z. This is fundamental for efficient inference.
    - **Active Paths:** A path between two variables is "active" if information can flow along it. Different configurations of nodes (chains, common causes,

common effects) and whether intermediate nodes are observed or unobserved determine if a path is active.

  - **Blocked Paths:** If a path is not active, it's blocked, implying conditional independence

Efficient Representation of Conditional Distributions

Efficiently representing Conditional Probability Distributions (CPDs) is crucial in AI, especially within probabilistic graphical models like Bayesian Networks. The challenge arises because a naive representation (e.g., a full table) can grow exponentially with the number of parent variables.

The Challenge of Naive CPD Representation

Consider a variable X with k possible values and n parent variables, P1,…,Pn, each with m possible values. A full Conditional Probability Table (CPT) for X would require k×mn entries. This exponential growth (mn) quickly becomes intractable as n increases. For example, if k=2 (binary variable) and each parent is binary (m=2), with just 10 parents, you'd need 2×210=2×1024=2048 entries. For 30 parents, it's 2×230, a massive number.

Efficient Representation Techniques

To overcome this, various specialized representations for CPDs have been developed:

1. Conditional Probability Tables (CPTs)
2. Noisy-OR/Noisy-MAX Distributions
3. Decision Trees/Decision Graphs (DTrees/DGraphs)
4. Context-Specific Independence (CSI)
5. Deterministic Relationships
6. Parametric Distributions (for Continuous Variables)
7. Neural Networks / Deep Learning Models:

Approximate Inference in Bayesian Networks

Approximate inference methods are indispensable in Bayesian Networks (BNs) when exact inference becomes computationally intractable

While exact inference algorithms (like Variable Elimination or Junction Tree Algorithm) guarantee precise probability distributions, their computational complexity often grows exponentially with the size of the network, especially for densely connected or "loopy" graphs

Why is Approximate Inference Needed?

1. **Computational Intractability**
2. Real-time Applications
3. Scalability
4. Learning in Large Networks

Approximate inference methods are indispensable in Bayesian Networks (BNs) when exact inference becomes computationally intractable. While exact inference algorithms (like Variable Elimination or Junction Tree Algorithm) guarantee precise probability distributions, their computational complexity often grows exponentially with the size of the network, especially for densely connected or "loopy" graphs.

Why is Approximate Inference Needed?

- **Computational Intractability:** For large, complex BNs with many variables, high-cardinality variables, or a high degree of connectivity (many loops), exact inference can be prohibitively slow and memory-intensive. The problem is often #P-hard.
- **Real-time Applications:** Many AI applications require quick probabilistic estimates (e.g., in robotics, real-time diagnostics, or decision support systems). Exact inference might not meet these time constraints.
- **Scalability:** To apply BNs to real-world problems with thousands or millions of variables (common in big data scenarios), approximate methods are the only feasible option.
- **Learning in Large Networks:** When learning the structure or parameters of a BN from data, inference steps are often part of the learning algorithm. Approximate inference makes learning tractable for larger models.

How Approximate Inference Works :

□ **Sampling-based Methods (Monte Carlo Methods):** These methods generate a large number of random samples from the network's distribution. The probabilities are then estimated by counting the frequency of events in these samples. The more samples generated, the better the approximation (by the law of large numbers).

□ **Variational Methods:** These methods reformulate the inference problem as an optimization problem. They try to find a simpler, "tractable" distribution that is "closest" to the true, intractable posterior distribution

Inference Techniques :

1. **Sampling-Based Methods**
    a. Prior Sampling
    b. Rejection Sampling
    c. Likelihood Weighting
    d. Markov Chain Monte Carlo (MCMC) - e.g., Gibbs Sampling
2. **Variational Methods**
    a. Variational Inference (VI)
    b. Loopy Belief Propagation (LBP)

Relational and First-Order Probability

"Relational and First-Order Probability" in AI refers to a field that combines the expressiveness of **First-Order Logic (FOL)** with the ability of **probability theory**

This area is often grouped under the umbrella of **Statistical Relational AI (StarAI)** or **Probabilistic Logic Programming (PLP)**

To overcome limitations of traditional probabilistic models (like Bayesian Networks or Markov Random Fields) and traditional logical AI :

1. ***Limitations of Propositional Probabilistic Models***
2. ***Limitations of Pure First-Order Logic***

1. **Limitations of Propositional Probabilistic Models:**

    - **Fixed Number of Variables:** Standard BNs operate on a fixed set of propositional variables (e.g., "IsSickJohn," "HasFeverMary"). They cannot easily represent domains where the number of entities or relationships varies dynamically.
    - **Lack of Generality:** To model multiple individuals (John, Mary, Alice, Bob), you need to create separate variables for each. This leads to a huge, repetitive model and doesn't capture general rules about how properties and relations behave across individuals.
    - **No Object Identity:** They don't explicitly represent individuals, types, and the relationships between them in a structured way.

2. **Limitations of Pure First-Order Logic:**

    - **Dealing with Uncertainty:** FOL is inherently deterministic. It's excellent for expressing "If X is true, then Y is true," but struggles with statements like "If a person has a fever, they are *likely* to be sick" or "Most birds can fly."
    - **Learning from Data:** While logical systems can be learned (e.g., Inductive Logic Programming), they often don't naturally integrate statistical learning from noisy data.

Bayes' Rule and its Use in Probabilistic Reasoning

Bayes' Rule is the cornerstone of probabilistic reasoning in AI, particularly for **belief updating**. It allows us to invert conditional probabilities, calculating the probability of a **cause (hypothesis)** given an **effect (evidence)**

1. Probabilistic Reasoning in AI

Probabilistic reasoning in AI is about making decisions and drawing inferences in the face of **uncertainty**. Unlike traditional logic that deals with true/false statements, probabilistic reasoning assigns degrees of belief (probabilities) to propositions and uses the laws of probability to update these beliefs as new evidence becomes available.

2. Bayes' Rule and its Use in Probabilistic Reasoning

Bayes' Rule Formula:

$P(H|E)=P(E)P(E|H)\cdot P(H)$

Where:

- $P(H|E)$: **Posterior Probability** - The probability of the hypothesis H being true, *after* observing the evidence E. This is what we want to calculate.

- $P(E|H)$: **Likelihood** - The probability of observing the evidence E, *given that* the hypothesis H is true. This is often easier to model or estimate (e.g., the probability of a positive test result if a disease is present).

- $P(H)$: **Prior Probability** - The initial probability of the hypothesis H being true, *before* any evidence E is observed. This represents our initial belief.

- $P(E)$: **Evidence Probability (Marginal Likelihood)** - The total probability of observing the evidence E, regardless of whether H is true or false. It acts as a normalizing constant.

  This can be expanded using the Law of Total Probability: $P(E)=\sum iP(E|Hi)\cdot P(Hi)$, where Hi are all possible hypotheses.

How Relational/First-Order Probability Formalisms integrate with Bayes' Rule:

1. **Probabilistic Relational Models (PRMs):**
   - PRMs define a probabilistic schema for a relational domain. They specify conditional dependencies between attributes of objects and relations between objects, often using statistical aggregation functions (e.g., `Grade(S,C)` depends on `Intelligence(S)` and the *average* `Intelligence` of `Students` in `Course C`)
2. **Markov Logic Networks (MLNs)**
   - MLNs combine First-Order Logic formulas with weights. Each weighted formula defines a "soft" constraint. An MLN defines a probability distribution over possible worlds (assignments of truth values to all ground predicates)
3. **Probabilistic Logic Programs (PLPs)**
   - PLPs (like ProbLog) extend logic programming with probabilistic facts and rules.