Module -4

PROBABILISTIC REASONING

4.1 Representing Knowledge in an uncertain domain

Representing knowledge in uncertain domains is a fundamental aspect of artificial intelligence, especially in fields like probabilistic reasoning, decision theory, and machine learning. Here are some common approaches:

1.Probabilistic Graphical Models (PGMs):

PGMs are a powerful framework for representing and reasoning under uncertainty. Bayesian networks and Markov networks are two main types of PGMs. They allow representing uncertain relationships between variables using probability distributions.

2.Fuzzy Logic:

Fuzzy logic allows representing uncertainty by assigning degrees of truth to propositions or conditions. It is particularly useful when dealing with vague or imprecise information.

3.**Possibility Theory:**

Similar to fuzzy logic, possibility theory deals with uncertainty by assigning degrees of possibility to propositions. It allows for reasoning about uncertainty in a flexible and intuitive way.

4.Decision Trees and Decision Networks:

Decision trees and decision networks are used to model decision-making processes under uncertainty. They represent decisions, uncertainties, and their potential outcomes in a structured manner.

5.Markov Decision Processes (MDPs):

MDPs are used to model decision-making in situations where outcomes are partially random and partially under the control of a decision-maker. They are widely used in reinforcement learning and planning.

6.Bayesian Inference:

Bayesian inference provides a principled framework for updating beliefs about uncertain quantities based on observed evidence. It is used extensively in various fields, including statistics, machine learning, and natural language processing.

7.Monte Carlo Methods:

Monte Carlo methods are a class of computational techniques used to estimate uncertain quantities by simulating random samples from a probability distribution. They are often used in probabilistic reasoning and decision-making problems.

8.Neural Networks for Uncertainty Estimation:

Recent advances in deep learning have led to the development of neural network architectures specifically designed to model and quantify uncertainty in predictions, such as Bayesian neural networks and variational autoencoders.

4.2 Semantics of Bayesian networks:

In artificial intelligence (AI), Bayesian networks are a powerful graphical model used to represent probabilistic relationships among a set of variables. They provide a compact and intuitive representation of uncertain knowledge, allowing for efficient reasoning and inference. Here's an overview of the semantics of Bayesian networks:

1.Directed Acyclic Graph (DAG):

The structure of a Bayesian network is defined by a directed acyclic graph, where nodes represent variables and directed edges represent probabilistic dependencies between variables. A node in the graph represents a random variable, and the edges represent direct influences or dependencies between these variables.

2.Conditional Probability Tables (CPTs):

Each node in a Bayesian network is associated with a conditional probability table (CPT) that quantifies the probabilistic relationship between that node and its parents in the graph. The CPT specifies the probability distribution of the node given the values of its parent nodes.

3.Conditional Independence:

One of the key semantic features of Bayesian networks is the concept of conditional independence. Two variables X and Y are conditionally independent given a set of variables Z if the probability distribution of X given Y and Z is the same as the probability distribution of X given Z alone.

Bayesian networks exploit conditional independence to factorize the joint probability distribution over all variables into a product of conditional probabilities.

4.Bayesian Inference:

Bayesian networks allow for efficient inference using probabilistic reasoning techniques such as Bayesian inference. Given evidence about certain variables, Bayesian inference calculates the posterior distribution of other variables in the network. This involves updating the probabilities of variables based on observed evidence using Bayes' theorem.

5.**Propagation of Probabilities**:

In Bayesian networks, probabilities can be propagated through the network using various inference algorithms, such as variable elimination, junction tree algorithms, or sampling methods like Markov Chain Monte Carlo (MCMC). These algorithms allow for efficient computation of marginal and conditional probabilities in the presence of evidence.

6.Decision Making:

Bayesian networks can also be used for decision-making under uncertainty. By incorporating decision nodes representing actions and utility nodes representing preferences or outcomes, Bayesian networks can model decision problems and facilitate optimal decision-making based on probabilistic reasoning.

7.Learning:

Bayesian networks can be learned from data using various techniques such as parameter learning and structure learning. Parameter learning involves estimating the parameters (CPTs) of the network from observed data, while structure learning involves discovering the underlying graph structure of the network from data.

4.3 Probabilistic reasoning over time:

Probabilistic reasoning over time, also known as temporal probabilistic reasoning, is an essential aspect of artificial intelligence, particularly in areas such as time-series analysis, sequential decision-making, and modeling dynamic systems. Here are some key techniques and approaches used for probabilistic reasoning over time in AI:

1.Hidden Markov Models (HMMs):

HMMs are probabilistic models widely used for modeling time-series data. They consist of a sequence of hidden states representing the underlying system dynamics and a sequence of observable emissions. HMMs are particularly useful for tasks such as speech recognition, gesture recognition, and bioinformatics.

2.Dynamic Bayesian Networks (DBNs):

DBNs extend Bayesian networks to model temporal dependencies in data. They consist of a series of Bayesian networks, each representing the state of the system at a specific time step, and temporal links between these networks to capture transitions over time. DBNs are employed in applications such as robotics, financial forecasting, and medical diagnosis.

3.Kalman Filters and Extended Kalman Filters:

Kalman filters are recursive algorithms used for estimating the state of a dynamic system from a series of noisy observations. Extended Kalman filters (EKFs) extend the Kalman filter to nonlinear systems by linearizing the dynamics. Kalman filters and EKFs are widely used in fields such as navigation, control systems, and signal processing.

4.Particle Filters:

Particle filters, also known as sequential Monte Carlo methods, are non-parametric Bayesian filters used for estimating the state of a dynamic system based on a sequence of observations. Particle filters are particularly useful in situations where the system dynamics are non-linear and non-Gaussian. They find applications in robotics, tracking, and localization.

5.Temporal Probabilistic Graphical Models (PGMs):

Temporal PGMs extend static PGMs (such as Bayesian networks and Markov networks) to model temporal dependencies explicitly. They allow for probabilistic reasoning over time by capturing dependencies between variables across different time steps. Temporal PGMs are employed in tasks such as activity recognition, environmental monitoring, and anomaly detection.

6.Recurrent Neural Networks (RNNs):

RNNs are a class of neural networks designed to process sequential data by maintaining a hidden state that captures temporal dependencies. Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) are popular variants of RNNs capable of modeling long-range dependencies in sequential data. RNNs are used in natural language processing, time-series prediction, and sequential decision-making.

7.Markov Decision Processes (MDPs):

MDPs are a framework for modeling decision-making in stochastic and dynamic environments. They consist of states, actions, transition probabilities, and rewards, allowing for probabilistic reasoning over time to find optimal policies. MDPs and their extensions (e.g., Partially Observable MDPs) are employed in reinforcement learning, robotics, and control systems.

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

- Bayes' rule
- Bayesian Statistics

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.

- $P(\neg A)$ = probability of a not happening event.
- $P(\neg A) + P(A) = 1$.

o Conditional probability:

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

o Let's suppose, we want 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:

o

o Where $P(A \wedge B)$= Joint probability of a and B

o $P(B)$= Marginal probability of B.

o If the probability of A is given and we need to find the probability of B, then it will be given as:

Bayes' theorem:

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:

$P(A \wedge B) = P(A|B) P(B)$ or

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

$P(A \wedge B) = P(B|A) P(A)$

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

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

Topology of Bayesian network

Here are the key components of the topology of a Bayesian network:

1.Nodes (Variables):

Each node in a Bayesian network represents a random variable or a feature in the domain being modeled. These variables can be discrete, continuous, or hybrid (mix of discrete and continuous). Nodes typically correspond to observable variables or latent variables (hidden variables).

2.Directed Edges (Arcs):

Directed edges between nodes represent probabilistic dependencies or causal relationships. An arrow from node A to node B indicates that node B is conditionally dependent on node A. In other words, the presence of an edge from node A to node B implies that the probability distribution of node B depends on the value of node A.

3.Conditional Probability Tables (CPTs):

Each node (except for root nodes) in a Bayesian network is associated with a conditional probability table (CPT). The CPT specifies the conditional probability distribution of the node given its parent nodes in the network. It quantifies how the probability of a node's value changes based on the values of its parents.

4.DAG Structure:

A Bayesian network has a Directed Acyclic Graph (DAG) structure, meaning there are no cycles in the network. This acyclic property ensures that the network does not contain feedback loops, making it suitable for efficient inference and reasoning.

5.Root Nodes and Leaf Nodes:

Root nodes are nodes in the Bayesian network that do not have any parents; they represent exogenous variables or variables whose values are determined externally. Leaf nodes are nodes that do not have any children; they represent variables whose values are directly observed or inferred.

6.Connectivity and Sparsity:

The connectivity of a Bayesian network refers to the degree to which nodes are connected to each other through directed edges. Sparsity refers to the property of having few connections or dependencies between nodes. Sparse networks are often preferred as they lead to more efficient inference and easier interpretation.

7.Bayesian Network Structure Learning:

Learning the topology of a Bayesian network involves determining the structure of the graph (i.e., the presence or absence of edges) from data. Structure learning algorithms use statistical methods and optimization techniques to identify the most probable network structure given observed data.

We want to represent the probability distribution of events:

– Burglary, Earthquake, Alarm, Mary calls and John calls

Directed acyclic graph

• Nodes = random variables Burglary, Earthquake, Alarm, Mary calls and John calls

• Links = direct (causal) dependencies between variables.

The chance of Alarm is influenced by Earthquake, The chance of John calling is affected by the Alarm

2. Local conditional distributions

• Relate variables and their parents

• The local probability information attached to each node takes the form of a conditional probability table (CPT).

• Each row in the CPT contains the conditional probability of each node value for conditioning case. A conditioning case is just the possible combination of values for the parent nodes.

• Each row must sum to 1, because the entries represent an exhaustive set of cases for the variable.

4.4 Inference in temporal models

Inference in temporal models in AI involves reasoning about the states of variables over time given observed data and possibly actions taken in the system. Temporal models capture the dynamics of systems where variables evolve over time, and inference in such models often requires considering dependencies and transitions between states across different time steps. Here are several techniques used for inference in temporal models:

1.Filtering:

Filtering involves estimating the current state of a system given all available observations up to the current time step. This is often done using techniques such as the Kalman filter for linear dynamical systems or particle filters for nonlinear and non-Gaussian systems.

2.Smoothing:

Smoothing extends filtering by estimating the entire trajectory of states over a sequence of time steps given all available observations. This is achieved by backward filtering techniques such as the

Rauch–Tung–Striebel (RTS) smoother for Kalman filters or particle smoothing methods for particle filters.

3.Prediction:

Prediction involves estimating the future states of a system given past observations up to the current time step. In temporal models, predictions can be made using forward filtering techniques, where the state estimates are propagated forward in time based on the system dynamics.

4.State Estimation in Hidden Markov Models (HMMs):

Inference in HMMs involves computing the posterior distribution over the hidden states of the model given the observed emissions. This can be efficiently performed using the forward-backward algorithm, which computes the forward and backward probabilities for each state at each time step.

5.Dynamic Bayesian Networks (DBNs):

In DBNs, inference involves reasoning about the joint distribution over variables at multiple time steps. This can be done using techniques such as the junction tree algorithm or approximate inference methods like particle filtering.

6.Temporal Probabilistic Graphical Models (PGMs):

Inference in temporal PGMs extends inference techniques from static PGMs to handle temporal dependencies. This often involves message-passing algorithms or variational inference techniques tailored to the temporal structure of the model.

7.Recurrent Neural Networks (RNNs):

Inference in RNNs involves updating the hidden states of the network over time based on sequential input data. This can be done using recurrent computation through time, where the hidden states are updated recursively according to the network's dynamics.

8.Temporal Logic and Temporal Reasoning:

In systems with temporal logic specifications, inference involves reasoning about the satisfaction of temporal properties over time. Techniques such as model checking or satisfiability modulo theories (SMT) solvers can be used for temporal reasoning tasks.

4.5 Hidden Markov models :

Hidden Markov Models (HMMs) are powerful probabilistic models used in artificial intelligence and machine learning for modeling sequential data, particularly when dealing with temporal dependencies and uncertainty. Here's an overview of HMMs in AI:

1.Basic Structure: A Hidden Markov Model consists of two main components:

- Hidden States: A sequence of hidden states representing the underlying, unobservable dynamics of the system. Each state is associated with a probability distribution.
- Observations: A sequence of observations representing the observable data generated by the hidden states. Each observation is generated from its corresponding hidden state based on an emission probability distribution.

2.Probabilistic Transitions:

- The transitions between hidden states are modeled using a transition probability matrix A, where Aij represents the probability of transitioning from state i to state j.
- The emission probabilities, representing the likelihood of each observation given the corresponding hidden state, are modeled using an emission probability matrix B.

3.Assumptions:

- Markovian property: The probability of transitioning to a particular state depends only on the current state, not on the entire history of states.
- Stationarity: The transition probabilities remain constant over time.
- Output Independence: The probability of observing a particular observation depends only on the current state, not on the entire history of observations.

4.Three Fundamental Problems:

- Evaluation: Given the model parameters compute the probability of observing a sequence of observations
- Decoding: Determine the most likely sequence of hidden states given the observations,

- Learning: Estimate the model parameters from a set of observed sequences, typically using the Baum-Welch algorithm, which is an instance of the Expectation-Maximization (EM) algorithm.

5.Applications:

- Speech Recognition: Modeling phonemes or words as hidden states and acoustic features as observations.
- Natural Language Processing: Part-of-speech tagging, named entity recognition, and language modeling.
- Bioinformatics: DNA sequence analysis, gene prediction, and protein structure prediction.
- Time-series Analysis: Financial market prediction, weather forecasting, and signal processing.

6.Extensions and Variants:

- Continuous HMMs: Where observations are continuous rather than discrete.
- Variable-Length HMMs: Allowing variable-length sequences of observations.
- Hidden Semi-Markov Models (HSMMs): Relaxing the Markovian assumption by allowing variable-duration hidden states.

4.6 Kalman Filter :

The Kalman Filter is a recursive algorithm used for estimating the state of a linear dynamic system from a series of noisy observations. It is a fundamental tool in artificial intelligence and control theory, particularly in applications involving state estimation, such as navigation, tracking, and signal processing. Here's an overview of the Kalman Filter in AI:

1.Basic Principles:

•The Kalman Filter operates on a linear dynamic system represented by the following equations:

2.Prediction Step:

•The Kalman Filter predicts the current state of the system based on the previous state estimate and the system dynamics.

3.Update Step:

•The Kalman Filter updates the state estimate based on the observed data.

•It computes the Kalman GainKt and uses it to update the state mean and covariance.

•The update equations are as follows:

Rt represents the observation noise covariance.

4.Initialization:

•The Kalman Filter requires an initial estimate of the state mean and covariance

5.Recursive Estimation:

•Once initialized, the Kalman Filter recursively predicts and updates the state estimate as new observations become available.

6.Optimal Estimation:

•The Kalman Filter provides an optimal estimation of the state in the minimum mean square error sense under the assumptions of linearity and Gaussian noise.

7.Extensions:

•Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) are extensions of the Kalman Filter that can handle non-linear systems by linearizing or approximating the system dynamics.