KALINGA UNIVERSITY

**KALINGA UNIVERSITY NAYA RAIPUR**

**Faculty of CS & IT**

**Programme: -BCAAIML**
**Course Name: - Machine Learning Techniques**
**Course Code: -BCAAIML501** **Sem : 5th**

**Unit-V**

## Graphical Models – Overview

Graphical models are a powerful framework for encoding **probabilistic relationships** among random variables using graphs. They combine **graph theory** and **probability theory** to visually and mathematically represent complex systems.

### 1. Types of Graphical Models

#### a. Bayesian Networks (Directed Graphical Models)

- **Definition**: Directed Acyclic Graphs (DAGs) where each node is a random variable and edges represent conditional dependencies.
- **Key Concepts**:
  - Conditional Probability Tables (CPTs)
  - Chain rule of probability
  - Topological ordering
- **Applications**:
  - Diagnosis systems (medical, mechanical)
  - Decision support systems

#### b. Markov Random Fields (Undirected Graphical Models)

- **Definition**: Undirected graphs representing mutual dependencies between variables.
- **Key Concepts**:
  - Markov properties
  - Cliques and potential functions
  - Factorization of joint distribution
- **Applications**:
  - Image segmentation
  - Computer vision

#### c. Factor Graphs

- **Definition**: Bipartite graphs showing how a global function (like a joint distribution) factors into local functions.
- **Used in**: Belief propagation algorithms, coding theory

### 2. Hidden Markov Models (HMMs)

- **Special type** of Bayesian network used for temporal data.
- **Components**:
  - States (hidden)
  - Observations (visible)
  - Transition and emission probabilities
- **Applications**:
  - Speech recognition
  - POS tagging
  - Gesture tracking

### 3. Markov Chain Monte Carlo (MCMC) Methods

- **Purpose**: Sampling from complex probability distributions where direct sampling is difficult.
- **Subtopics**:
  - **Markov Chains**: Sequence of random variables where the future state depends only on the current state.
  - **Monte Carlo Sampling**: Using random samples to approximate expectations.

### 4. Sampling Techniques in MCMC

#### a. Proposal Distribution

- Used to propose the next state in MCMC.
- Should be easy to sample from.
- Examples: Gaussian, uniform

#### b. Metropolis-Hastings Algorithm

- Accepts or rejects new samples based on an acceptance ratio.
- Allows for flexibility in choosing the proposal distribution.

#### c. Gibbs Sampling

- A special case of MCMC where each variable is sampled conditionally on the others.

### 5. Tracking Methods Using Graphical Models

- **Used to estimate dynamic system states over time.**
- **Examples**:
  - **Particle Filters**: Use a set of samples (particles) to represent the posterior distribution over time.
  - **Kalman Filters**: Linear-Gaussian models for continuous state estimation.
  - **HMM-based tracking**: For discrete state tracking with uncertain observations.

Here is a detailed explanation of **Markov Chain Monte Carlo (MCMC) Methods**, with clear sub-points for easy understanding:

## Markov Chain Monte Carlo (MCMC) Methods

MCMC methods are a class of algorithms used to **sample from complex probability distributions** when direct sampling is difficult. They use **Markov chains** to generate samples that can approximate the desired distribution through repeated random sampling.

### Why Use MCMC?

- To perform **inference** in **high-dimensional or complex probabilistic models**.
- When **analytical solutions are intractable**.
- To estimate **posterior distributions** in **Bayesian statistics**.

### Core Concepts

#### a. Markov Chain

- A sequence of random states X1,X2,…X_1, X_2, \dots where the next state depends **only on the current state** (Markov property).
- Characterized by a **transition probability matrix**.

#### b. Monte Carlo Sampling

- Involves using **random samples** to compute numerical estimates of functions (e.g., integrals, expectations).

#### c. Stationary Distribution

- The distribution to which the Markov chain **converges over time**.
- In MCMC, this is the **target distribution** we want to sample from.

### Key MCMC Algorithms

#### a. Metropolis-Hastings Algorithm

- Proposes a new sample x′x' from a **proposal distribution** q(x′|x)q(x'|x).
- Accepts the new sample with a probability:

  α=min(1,p(x′)q(x|x′)p(x)q(x′|x))\alpha = \min\left(1, \frac{p(x')q(x|x')}{p(x)q(x'|x)}\right)

- Ensures samples eventually reflect the **target distribution** p(x)p(x).

#### b. Gibbs Sampling

- Special case of MCMC for **multivariate distributions**.
- Iteratively samples each variable from its **conditional distribution**, holding the others fixed.

- Useful when conditionals are easier to sample from.

**Proposal Distribution**

- A function used to propose the next candidate state in MCMC.
- Common choices:
  - **Gaussian**: For continuous state space
  - **Uniform**: For bounded sampling
- Affects the **efficiency and convergence rate** of the sampler.

**Applications of MCMC**

- **Bayesian inference** (e.g., computing posterior distributions)
- **Machine learning** (e.g., training probabilistic models)
- **Statistical physics**
- **Econometrics**
- **Computational biology**

**Advantages**

- Handles **high-dimensional** and **non-convex** spaces.
- Does **not require normalization constants** for probability distributions.
- Flexible with **different types of models**.

**Limitations**

- **Convergence** can be slow or hard to detect.
- Needs many samples for **accurate estimates**.
- Poor choice of proposal distribution can result in **inefficient sampling**.

**Sampling – Overview**

**Sampling** is the process of selecting random values from a **probability distribution**. It's essential in situations where:

- The distribution is **complex or high-dimensional**.
- **Exact inference** is computationally expensive or infeasible.

Sampling helps **approximate expectations**, **marginals**, and **posterior distributions** in various probabilistic models.

**Importance of Sampling**

- Used in **Monte Carlo methods** for numerical estimation.
- Essential in **Bayesian inference** for estimating posterior distributions.
- Supports **learning and inference** in graphical models (e.g., MCMC, Gibbs sampling).

**Types of Sampling Techniques**

**a. Simple Random Sampling**

- Every data point has an **equal chance** of being selected.
- Used in basic statistical applications.

**b. Importance Sampling**

- Samples are drawn from a **proposal distribution** rather than the target.
- Each sample is **weighted** to correct for the difference:

  w(x)=p(x)q(x)w(x) = \frac{p(x)}{q(x)}

  where p(x)p(x) is the target and q(x)q(x) is the proposal distribution.

**c. Rejection Sampling**

- Samples from a proposal distribution q(x)q(x).
- Accepts the sample with a probability proportional to:

  p(x)Mq(x)\frac{p(x)}{Mq(x)}

  where MM is a constant such that p(x)≤Mq(x)p(x) \leq Mq(x) for all xx.

**d. Markov Chain Sampling**

- Generates samples by building a **Markov chain** that converges to the target distribution.
- Basis for **MCMC methods** (Metropolis-Hastings, Gibbs sampling).

**Sampling in Graphical Models**

Sampling helps estimate:

- **Marginal probabilities** over nodes
- **Posterior distributions** given evidence
- **Predictions** in Bayesian networks and Markov random fields

Used in algorithms like:

- **Particle filtering** (sequential sampling for time-series models)
- **Approximate inference** in loopy networks

**Evaluation of Sampling Methods**

Criteria include:

- **Efficiency**: Speed of generating high-quality samples.
- **Convergence**: How quickly the sample distribution matches the target.
- **Variance**: Lower variance estimates are better.

**Applications**

- **Bayesian Machine Learning**
- **Probabilistic Reasoning**
- **Robotics (Localization, SLAM)**
- **Computer Vision (Tracking, Image Modeling)**

## Proposal Distribution – Overview

A **proposal distribution** is a **probability distribution** used to **suggest candidate samples** in MCMC algorithms like **Metropolis-Hastings**. It plays a crucial role in **guiding the exploration** of the state space.

### Purpose of Proposal Distribution

- To generate a **candidate point** x′x' from the current state xx.
- Used in the acceptance-rejection decision to ensure the samples eventually reflect the **target distribution** p(x)p(x).

### Role in Metropolis-Hastings Algorithm

Given:

- Current state xx
- Proposed state x′∼q(x′|x)x' \sim q(x'|x)
- Acceptance probability:

  α=min(1,p(x′)q(x|x′)p(x)q(x′|x))\alpha = \min\left(1, \frac{p(x')q(x|x')}{p(x)q(x'|x)}\right)

Here:

- p(x)p(x): Target distribution
- q(x′|x)q(x'|x): Proposal distribution
- Determines whether to **accept or reject** the proposed sample

### Types of Proposal Distributions

#### a. Symmetric Proposal

- q(x′|x)=q(x|x′)q(x'|x) = q(x|x')
- Simplifies the acceptance ratio to:

  α=min(1,p(x′)p(x))\alpha = \min\left(1, \frac{p(x')}{p(x)}\right)

- Example: **Gaussian** centered at current state

#### b. Asymmetric Proposal

- q(x′|x)≠q(x|x′)q(x'|x) \neq q(x|x')
- More flexible, but requires full ratio in acceptance probability
- Useful when target distribution is skewed

**Design Considerations**

A good proposal distribution should:

- Be **easy to sample from**
- Ensure **adequate exploration** (not too narrow or too wide)
- Provide **high acceptance rates**
- Allow the Markov chain to **mix well** (i.e., reach target distribution quickly)

Poor design can lead to:

- **Slow convergence**
- **High autocorrelation** between samples
- **Biased or inefficient** sampling

**Examples of Proposal Distributions**

| Type | Description | Example |
|---|---|---|
| Uniform | Random jump within a fixed window | ( q(x' |
| Gaussian | Small local step around current state | ( q(x' |
| Adaptive | Adjusts based on previous samples | Increases efficiency |

**Applications**

- **Metropolis-Hastings sampling**
- **Hamiltonian Monte Carlo (HMC)**: Uses gradients for smarter proposals
- **Particle filters** in tracking and localization

**Markov Chain Monte Carlo (MCMC) – Overview**

**MCMC** is a family of algorithms used to generate **samples from a complex probability distribution** by constructing a **Markov chain** that has the desired distribution as its **stationary distribution**.

🔍 **Goal**: To approximate **expectations or probabilities** when direct sampling is not feasible due to high-dimensional or unknown normalizing constants.

**Key Concepts**

**a. Markov Chain**

- A sequence of random variables X1,X2,…X_1, X_2, \dots where the probability of the next state depends **only on the current state**:

  P(Xn+1=x′|Xn=x)P(X_{n+1} = x' | X_n = x)

**b. Monte Carlo Method**

- A numerical method using **random sampling** to estimate mathematical quantities (like integrals or expectations).

**c. Stationary Distribution**

- A distribution π\pi such that:

  π(x′)=∑xπ(x)P(x→x′)\pi(x') = \sum_x \pi(x) P(x \rightarrow x')

- The **target distribution** we want to sample from.

**Why Use MCMC?**

- Direct sampling from complex posteriors is difficult.
- **Exact inference** is intractable in many probabilistic models.
- Allows **approximate inference** in **Bayesian statistics**, **graphical models**, and more.

**Common MCMC Algorithms**

**a. Metropolis Algorithm**

- Uses a **symmetric proposal distribution** q(x′|x)=q(x|x′)q(x'|x) = q(x|x')
- Accepts new state x′x' with probability:

  α=min(1,p(x′)p(x))\alpha = \min\left(1, \frac{p(x')}{p(x)}\right)

**b. Metropolis-Hastings Algorithm**

- Generalized form with **asymmetric proposal distribution**
- Accepts x′x' with probability:

  α=min(1,p(x′)q(x|x′)p(x)q(x′|x))\alpha = \min\left(1, \frac{p(x')q(x|x')}{p(x)q(x'|x)}\right)

**c. Gibbs Sampling**

- Special case where each variable is sampled from its **conditional distribution**, one at a time.
- Very useful when conditional distributions are known and easy to sample from.

**Proposal Distribution**

- Suggests a new candidate state.
- Should balance **exploration** (covering the space) and **efficiency** (high acceptance rate).
- Examples:
  - Gaussian around current state
  - Uniform within a range

**Steps in MCMC Sampling (Metropolis-Hastings)**

1. Initialize at x0x_0
2. Propose x′∼q(x′|xt)x' \sim q(x'|x_t)
3. Calculate acceptance probability α\alpha
4. Accept or reject the proposal
5. Repeat for desired number of iterations

**Convergence and Burn-in**

- The Markov chain takes time to **converge to the target distribution**.
- Initial samples (called **burn-in** period) are discarded to reduce bias.

**Applications of MCMC**

- **Bayesian statistics** (posterior estimation)
- **Machine learning** (parameter estimation in probabilistic models)
- **Computational biology** (gene network modeling)
- **Computer vision** (image segmentation, tracking)
- **Physics** (simulation of particle systems)

**Advantages**

- Handles **high-dimensional** and **complex distributions**
- No need for **normalizing constants**
- Flexible for a wide range of models

**Limitations**

- May converge **slowly**
- Sensitive to choice of **proposal distribution**
- Requires **diagnostics** to ensure proper convergence
- Samples are often **correlated**

## Graphical Models – Overview

Graphical Models are **probabilistic models** that use **graphs** to represent the **conditional dependencies** between random variables.
They combine elements of **graph theory** and **probability theory**, making complex relationships easier to visualize and compute.

**Purpose and Importance**

- Simplify **joint probability distributions**.
- Help in **reasoning under uncertainty**.
- Enable **efficient inference and learning** in complex systems.
- Common in **machine learning**, **computer vision**, **natural language processing**, and **bioinformatics**.

**Types of Graphical Models**

**a. Bayesian Networks (BNs)**

- **Directed Acyclic Graphs (DAGs)**.
- Represent **causal relationships** or **conditional dependencies**.
- Each node = random variable.
- Each edge = conditional dependency.

**Properties**:

- Local Conditional Probability Distributions (CPDs)
- Joint distribution factorizes as:

  P(X1,...,Xn)=∏i=1nP(Xi|Parents(Xi))P(X_1, ..., X_n) = \prod_{i=1}^n P(X_i | \text{Parents}(X_i))

**Example Use Cases**:

- Medical diagnosis
- Decision-making systems

**b. Markov Random Fields (MRFs) or Undirected Graphical Models**

- **Undirected graphs**.
- Represent **mutual dependencies** or **spatial relationships**.
- Each node = variable; edges show dependence without direction.

**Factorization**:

P(X1,...,Xn)=1Z∏C∈CϕC(XC)P(X_1, ..., X_n) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \phi_C(X_C)

Where ϕC\phi_C is a potential function over clique CC, and ZZ is the normalization constant.

**Applications**:

- Image denoising
- Spatial modeling

**c. Factor Graphs**

- Bipartite graphs with **variable nodes** and **factor nodes**.
- Used to express the **factorization** of a global function.
- Simplifies implementation of **message passing algorithms**.

**Key Concepts**

**a. Conditional Independence**

- Graphs encode independence assumptions.
- **Bayesian Network**: d-separation

- **MRF**: Graph separation

**b. Inference**

- Computing **marginal** or **conditional** probabilities.
- **Exact** inference: Variable elimination, belief propagation.
- **Approximate** inference: MCMC, Variational inference.

**c. Learning**

- **Parameter Learning**: Estimating the CPDs or potential functions.
- **Structure Learning**: Learning the graph structure from data.

**Hidden Markov Models (HMMs) – A Special Case**

- Temporal graphical model.
- States are hidden; observations are visible.
- Used in speech, tracking, and sequence modeling.

**Applications of Graphical Models**

- **Medical Diagnosis** (Bayesian Networks)
- **Computer Vision** (MRFs, CRFs)
- **Natural Language Processing** (HMMs, CRFs)
- **Speech Recognition** (HMMs)
- **Genomics** (Graphical models for gene networks)

**Advantages**

- Modular representation of complex systems
- Encodes both **qualitative structure** and **quantitative relationships**
- Scales well to large problems with **local dependencies**

**Bayesian Networks (BNs) – Overview**

A **Bayesian Network** is a **probabilistic graphical model** that represents a **set of variables and their conditional dependencies** using a **Directed Acyclic Graph (DAG)**.

Each node in the graph corresponds to a random variable, and each directed edge indicates a **direct influence** or **causal relationship**.

**Components of a Bayesian Network**

| Component | Description |
|---|---|
| **Nodes** | Random variables (discrete or continuous) |
| **Edges** | Directed edges indicating **causal or conditional** |

| Component | Description dependence |
|---|---|
| **Conditional Probability Tables (CPTs)** | Quantify the effect of parents on each variable |

**Factorization of Joint Probability**

If the network has nodes X1,X2,...,XnX_1, X_2, ..., X_n, the joint distribution is factorized as:

P(X1,X2,...,Xn)=∏i=1nP(Xi|Parents(Xi))P(X_1, X_2, ..., X_n) = \prod_{i=1}^n P(X_i \mid \text{Parents}(X_i))

This reduces computation by exploiting **conditional independence**.

**Key Concepts**

**a. Conditional Independence**

- A node is conditionally independent of its **non-descendants**, given its **parents**.
- This allows efficient **inference** and **learning**.

**b. d-separation**

- A graphical criterion to determine conditional independence.
- If two nodes are **d-separated** given a set of nodes, they are conditionally independent.

**c. Inference in Bayesian Networks**

- **Goal**: Compute marginal or conditional probabilities.
- **Methods**:
  - Exact: Variable Elimination, Belief Propagation
  - Approximate: MCMC, Likelihood Weighting

**Learning in Bayesian Networks**

**a. Parameter Learning**

- Estimating **CPTs** from data if structure is known.
- Methods: Maximum Likelihood Estimation (MLE), Bayesian Estimation

**b. Structure Learning**

- Discovering the DAG structure from data.
- Approaches:
  - **Score-based** (e.g., BIC, AIC)

- **Constraint-based** (e.g., PC Algorithm)
- **Hybrid methods**

**Example**

**Simple Bayesian Network:**

Rain → WetGrass ← Sprinkler

- P(WetGrass|Rain,Sprinkler)P(WetGrass | Rain, Sprinkler)
- P(Rain),P(Sprinkler)P(Rain), P(Sprinkler)

You can factor:

P(R,S,W)=P(R)·P(S)·P(W|R,S)P(R, S, W) = P(R) \cdot P(S) \cdot P(W \mid R, S)

**Applications of Bayesian Networks**

- **Medical diagnosis** (e.g., predicting disease from symptoms)
- **Spam filtering**
- **Sensor networks**
- **Fraud detection**
- **Genetics and bioinformatics**

**Advantages**

- Compact representation of joint distribution
- Encodes causal relationships
- Supports both qualitative and quantitative reasoning
- Efficient inference with sparse graphs

**Markov Random Fields (MRFs) – Overview**

A **Markov Random Field** (also called an **undirected graphical model**) is a probabilistic model representing the **joint distribution** of a set of random variables using an **undirected graph**.

Unlike Bayesian networks (which are directed), MRFs are **undirected** and are better suited for **symmetric relationships** like those found in image processing or spatial data.

**Structure of MRF**

| Element | Description |
|---|---|
| **Nodes** | Random variables (e.g., pixels, labels) |
| **Edges** | Undirected links between variables |
| **Cliques** | Fully connected subsets of nodes (local groupings) |
| **Potentials** | Functions assigned to cliques to represent interactions |

**Markov Properties**

MRFs rely on **conditional independence** among variables, defined by the graph structure:

**a. Pairwise Markov Property**

If two nodes are not connected, they are conditionally independent given all other nodes.

**b. Local Markov Property**

A node is conditionally independent of all other nodes given its neighbors.

**c. Global Markov Property**

Any two subsets of nodes are conditionally independent if a separating set blocks all paths between them.

**Joint Probability in MRFs**

The joint distribution of an MRF is represented using **clique potentials**:

P(X1,X2,…,Xn)=1Z∏C∈CϕC(XC)P(X_1, X_2, \ldots, X_n) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \phi_C(X_C)

Where:

- ϕC(XC)\phi_C(X_C): Potential function over clique CC
- ZZ: Partition function (normalizing constant):

  Z=∑X∏C∈CϕC(XC)Z = \sum_X \prod_{C \in \mathcal{C}} \phi_C(X_C)

**Inference in MRFs**

**a. Exact Inference:**

- Variable elimination
- Junction tree algorithm

**b. Approximate Inference:**

- Gibbs Sampling (MCMC)
- Loopy Belief Propagation
- Mean Field Methods

**Learning in MRFs**

**a. Parameter Learning**

- Estimating potential function parameters

- Methods: Maximum Likelihood, Pseudo-likelihood (used when computing ZZ is hard)

**b. Structure Learning**

- Learning the graph structure from data
- More challenging than parameter learning

**Example: Image Denoising**

- Each pixel = node
- Edge = dependency between neighboring pixels
- Goal = infer true pixel values given noisy observations
- **MRFs model smoothness** by encouraging nearby pixels to take similar values

**Applications of MRFs**

| Domain | Application |
|---|---|
| **Computer Vision** | Image segmentation, stereo vision |
| **Natural Language** | POS tagging, sequence labeling |
| **Statistical Physics** | Modeling particle interactions |
| **Spatial Statistics** | Weather modeling, geology |

**Hidden Markov Models (HMMs) – Overview**

A **Hidden Markov Model** is a **statistical model** used to describe systems that are **Markov processes with hidden (unobservable) states**.

It is especially useful for **time-series** or **sequential data**, where:

- The system evolves over time,
- We can observe only the output, not the internal (hidden) states.

**Components of an HMM**

An HMM is defined by the following parameters:

| Component | Description |
|---|---|
| **States** (hidden) | Set of possible internal states S={s1,s2,… }S = \{s_1, s_2, \ dots\} |
| **Observations** | Set of possible observations O={o1,o2,… }O = \{o_1, o_2, \ dots\} |
| **Transition Probabilities** AA | ( A_{ij} = P(s_j |
| **Emission Probabilities** BB | ( B_j(k) = P(o_k |
| **Initial Probabilities** π\pi | πi=P(si)\pi_i = P(s_i): Probability of starting in state sis_i |

**Graphical Structure**

An HMM is a **Dynamic Bayesian Network**, typically visualized as:

Hidden states: S1 → S2 → S3 → ... → ST
Observed outputs: O1 O2 O3 OT

Each hidden state emits an observable output.

**Key Assumptions**

1. **Markov Property**: The current state depends only on the **previous state** (first-order Markov process).
2. **Output Independence**: The observation at time tt depends only on the **current hidden state**.

**Common Problems Solved Using HMMs**

**a. Evaluation (Likelihood)**

**Problem**: Given the model λ\lambda and observation sequence OO, compute P(O|λ)P(O | \ lambda).
**Solution**: **Forward Algorithm**

**b. Decoding (Most probable state sequence)**

**Problem**: Given observations OO, find the most likely sequence of hidden states.
**Solution**: **Viterbi Algorithm**

**c. Learning (Training)**

**Problem**: Given observations, adjust parameters A,B,πA, B, \pi to best fit the data.
**Solution**: **Baum-Welch Algorithm** (a type of Expectation-Maximization)

**Tracking Methods – Overview**

**Tracking** refers to the process of estimating the **state** of a moving or changing object over time, using **noisy and partial observations**. It is widely used in:

- Computer Vision (object tracking)
- Robotics (localization)
- Signal Processing (target tracking)
- Natural Language Processing (sequence modeling)

**Key Concepts in Tracking**

| Term | Description |
|---|---|
| **State** | The true condition of the system at time tt (e.g., position, speed) |

| Term | Description |
|---|---|
| **Observation** | Measured data related to the state, often noisy |
| **Motion Model** | Describes how the state evolves over time |
| **Observation Model** | Links the observations to the underlying state |
| **Prediction** | Estimate the next state |
| **Correction** | Adjust estimate based on new observation |

### Tracking as a Probabilistic Problem

Tracking involves estimating the **posterior probability**:

P(xt|z1:t)P(x_t \mid z_{1:t})

Where:

- xtx_t = state at time tt
- z1:tz_{1:t} = all observations up to time tt

### Common Tracking Methods

#### a. Kalman Filter

- Assumes **linear system dynamics** and **Gaussian noise**.
- Maintains state using **mean** and **covariance**.
- Suitable for smooth and predictable motion.

**Steps**:

1. Predict next state using motion model.
2. Update prediction with new observation.

**Used in**:

- Navigation systems
- Motion tracking with sensors

#### b. Extended Kalman Filter (EKF)

- For **non-linear systems**.
- Linearizes using Jacobians.

#### c. Particle Filter (Sequential Monte Carlo)

- Represents posterior using a set of **weighted samples (particles)**.
- Handles **non-linear, non-Gaussian** systems.
- Each particle is a hypothesis of the state.

**Steps**:

1. Sample particles from motion model
2. Update weights based on observation likelihood
3. Resample particles

**Used in**:

- Object tracking in cluttered environments
- Robot localization (Monte Carlo Localization)

**d. Hidden Markov Models (HMMs)**

- Discrete-state tracking.
- Uses **Viterbi algorithm** to find the most probable state sequence.
- Suitable for applications like speech or activity recognition.

**e. Optical Flow (Visual Tracking)**

- Estimates pixel movement across frames.
- Used in video tracking.

**f. Kalman + MCMC Hybrid Approaches**

- Combine Kalman filters with MCMC or particle filtering.
- Useful when tracking involves multimodal or ambiguous data.

**Applications of Tracking Methods**

| Domain | Example Use |
|---|---|
| **Computer Vision** | Object tracking in video, face detection |
| **Robotics** | Robot path estimation, SLAM |
| **Surveillance** | People or vehicle tracking in real-time |
| **Augmented Reality** | Tracking headset or controller position |
| **Natural Language** | Tracking context or hidden states in sequence |

**Challenges in Tracking**

- **Occlusion**: Object gets hidden
- **Drift**: Small errors accumulate
- **Appearance changes**: Due to lighting or orientation
- **Real-time constraints**

**Tracking Using Graphical Models**

- Graphical models like **Dynamic Bayesian Networks (DBNs)** can model the tracking process.
- **Nodes** represent states and observations over time.
- **Inference** (e.g., filtering, smoothing) is performed to estimate hidden states.