UNIT-V

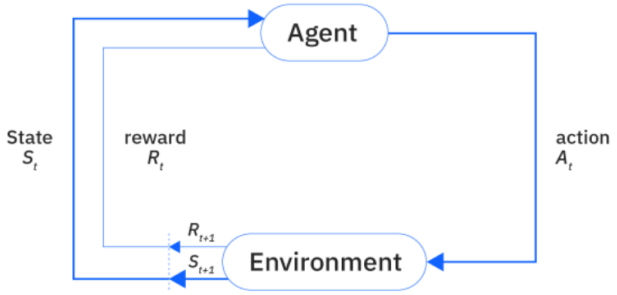
Reinforcement Learning; Overview of reinforcement learning, Getting Lost Example.

Markov Chain Monte Carlo Methods: Sampling, Proposal Distribution, Markov Chain Monte Carlo.

Graphical Models: Bayesian Networks, Markov Random Fields, Hidden Markov Models, Tracking Methods.

Reinforcement Learning:

- Reinforcement learning (RL) is a type of machine learning where an agent learns to make decisions by interacting with an environment.
- The agent receives feedback in the form of rewards or penalties based on the actions it takes, and its goal is to maximize the cumulative reward over time.



Key Components of Reinforcement Learning

1. **Agent**:
   - The learner or decision-maker that interacts with the environment.
2. **Environment**:
   - The external system the agent interacts with. It provides feedback based on the agent's actions.
3. **State**:
   - A representation of the current situation of the environment. The agent perceives the environment through states.
4. **Action**:
   - The set of all possible moves the agent can make in the environment.
5. **Reward**:
   - Feedback from the environment based on the agent's actions. Positive rewards incentivize desirable actions, while negative rewards (or penalties) discourage undesirable actions.
6. **Policy**:
   - A strategy used by the agent to determine the next action based on the current state. It can be deterministic or stochastic.
7. **Value Function**:
   - A function that estimates the expected cumulative reward of states or state-action pairs, helping the agent to make decisions that maximize long-term rewards.

The Learning Process

1. **Exploration**:
    - The agent tries out different actions to discover their effects and gather information about the environment.
2. **Exploitation**:
    - The agent uses its knowledge to choose actions that it believes will maximize the reward.
3. **Balance**:
    - Effective RL requires balancing exploration and exploitation to ensure the agent learns the optimal policy.

Algorithms in Reinforcement Learning

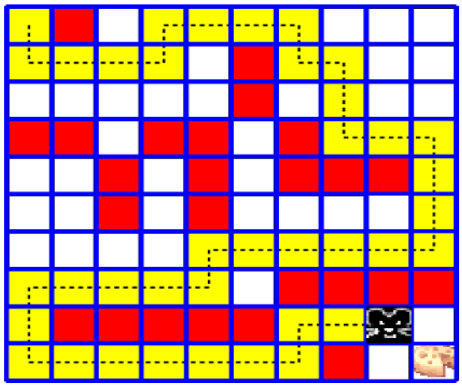
1. **Q-Learning**:
    - A model-free algorithm where the agent learns a value function Q(s,a), which represents the expected utility of taking action a in state s and following the optimal policy thereafter.
2. **SARSA (State-Action-Reward-State-Action)**:
    - Similar to Q-Learning, but updates the Q-value based on the action actually taken, considering the policy followed by the agent.

Getting Lost Example:

- "getting lost" example using reinforcement learning to understand how an agent learns to navigate an environment, avoid pitfalls, and reach its goal.
- We can imagine a scenario where an agent (like a robot) is placed in a maze and needs to find its way to the exit.

Scenario: Robot Navigating a Maze

1. **Environment**:
    - The maze consists of a grid with walls, open spaces, and an exit.
    - The robot starts at a random position and must find the exit.
2. **State**:
    - The current position of the robot in the maze, represented by coordinates (x, y).
3. **Actions**:
    - The robot can move up, down, left, or right.
4. **Rewards**:
    - Positive reward for reaching the exit.
    - Negative reward for hitting a wall.



Applications of Reinforcement Learning:

Reinforcement learning is a powerful approach to building intelligent systems that can adapt and improve through experience, opening up possibilities across a wide range of applications.

1. **Game Playing**: RL agents have achieved superhuman performance in games like chess, Go, and video games (e.g., AlphaGo, OpenAI Five).
2. **Robotics**: Training robots to perform complex tasks such as walking, grasping objects, and navigating environments.
3. **Autonomous Vehicles**: Learning to drive safely and efficiently in various traffic conditions.
4. **Healthcare**: Optimizing treatment strategies, personalized medicine, and managing clinical trials.
5. **Finance**: Algorithmic trading, portfolio management, and risk assessment.

Markov Chain Monte Carlo Methods:

- Markov Chain Monte Carlo (MCMC) methods are a class of algorithms used to sample from complex probability distributions, especially when direct sampling is difficult.
- These methods are widely used in Bayesian statistics, computational physics, and other fields where dealing with high-dimensional integrals is necessary.

Key Concepts of MCMC Methods:

1. **Markov Chain**:

- A sequence of random variables where the next state depends only on the current state (the Markov property).
- The chain has a stationary distribution that it converges to over time.

2. **Monte Carlo**:

- A technique that uses random sampling to estimate numerical results.
- In MCMC, Monte Carlo methods are used to generate samples from the probability distribution.

How MCMC Works:

1. **Initialization**:
    - Start with an initial state (or set of states) from the target distribution.
2. **Iteration**:
    - Propose a new state based on a proposal distribution.
    - Accept or reject the new state based on a criterion (e.g., Metropolis-Hastings algorithm)

3. **Convergence**:

    - After many iterations, the distribution of the states will approximate the target distribution.

Common Algorithms

1. **Metropolis-Hastings Algorithm**:
    - Proposes new states and accepts or rejects them based on the acceptance ratio
    - Widely used for its simplicity and flexibility.
2. **Gibbs Sampling**:

    - Samples each variable in turn, conditional on the current values of the other variables.
    - Useful when the conditional distributions are easier to sample from.

Applications of MCMC

1. **Bayesian Inference**:
    - Estimating posterior distributions of parameters when the likelihood and prior are known.
    - Useful for hierarchical models and complex data structures.
2. **Statistical Physics**:

    - Simulating the behavior of physical systems at the atomic or molecular level.
    - Estimating properties like magnetization or phase transitions.

3. **Machine Learning**:

    - Training models with complex likelihood functions.
    - Bayesian neural networks and other probabilistic models.

4. **Ecology and Evolutionary Biology**:

    - Estimating parameters of population dynamics models.
    - Studying the evolution of traits under different selection pressures.

Sampling:

Sampling is a technique used to select a subset of data from a larger population, allowing for the analysis and inference of population characteristics without examining the entire dataset.

Types of Sampling

1. **Probability Sampling**:
    - **Description**: Every member of the population has a known, non-zero chance of being selected.
    - **Examples**:
        - **Simple Random Sampling**: Every member of the population has an equal chance of being selected.
        - **Systematic Sampling**: Selects every k-th member from a list after a random start.
        - **Stratified Sampling**: Divides the population into strata (groups) and samples from each stratum.
        - **Cluster Sampling**: Divides the population into clusters and randomly selects entire clusters.
2. **Non-Probability Sampling**:
    - **Description**: Not every member of the population has a known or equal chance of being selected.
    - **Examples**:
        - **Convenience Sampling**: Samples are selected based on their availability or ease of access.
        - **Judgmental (Purposive) Sampling**: Samples are selected based on the researcher's judgment.
        - **Quota Sampling**: Ensures representation by selecting samples to meet certain quotas.
        - **Snowball Sampling**: Current subjects recruit future subjects from their acquaintances.

Proposal Distribution:

- A proposal distribution is a fundamental component in Markov Chain Monte Carlo (MCMC) methods.
- It is used to generate new candidate samples from a target probability distribution, especially when direct sampling is not feasible.
- A proposal distribution, denoted as $q(x'|x)$, is a probability distribution used to propose new candidate states x' given the current state x.
- The new candidate state is then accepted or rejected based on a criterion designed to ensure that the sequence of samples converges to the target distribution $\pi(x)$.

Markov Chain Monte Carlo Algorithms:

Metropolis-Hastings Algorithm:

- **Description**: A general algorithm that generates a candidate sample from a proposal distribution and accepts or rejects it based on an acceptance probability.
- **Process**:
    - Initialize at a state x_0.
    - Generate a candidate state x' from the proposal distribution $q(x'|x)$.
    - Calculate the acceptance probability:

$$\alpha = \min\left(1, \frac{\pi(x')q(x \mid x')}{\pi(x)q(x' \mid x)}\right)$$

4. Accept x′ with probability α, otherwise, stay at x.

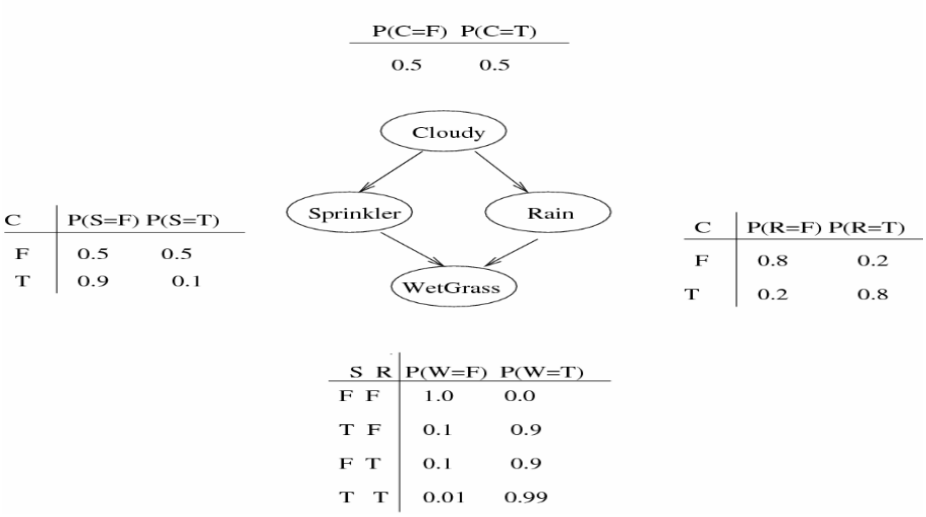
- **Use Case**: Widely applicable and flexible for various target distributions.

Gibbs Sampling:

- **Description**: Samples each variable in turn from its conditional distribution given the current values of the other variables.
- **Process**:
    1. Initialize all variables.
    2. Sample each variable xi from p(xi|other variables).
    3. Repeat until convergence.
- **Use Case**: Effective when conditional distributions are easier to sample from.
- **Example**: Ideal for Bayesian networks and hierarchical models.

Graphical Models:

- Graphical models are a powerful framework for representing complex dependencies among variables in a visual and mathematical way.



$$p(C, S, R, W) = p(C)p(S|C)p(R|C)p(W|S, R)$$

Bayesian Networks:

- Bayesian Networks (BNs) are a type of probabilistic graphical model that uses directed acyclic graphs (DAGs) to represent a set of variables and their conditional dependencies.
- They are particularly powerful for modeling complex systems where understanding the relationships between variables is crucial.

Joint Probability:

- Joint probability is a probability of two or more events happening together. For example, the joint probability of two events A and B is the probability that both events occur, P(A∩B).
  P(A ∩ B) = P(A) · P(B)
  P(A ∩ B) = P(A | B) · P(B)

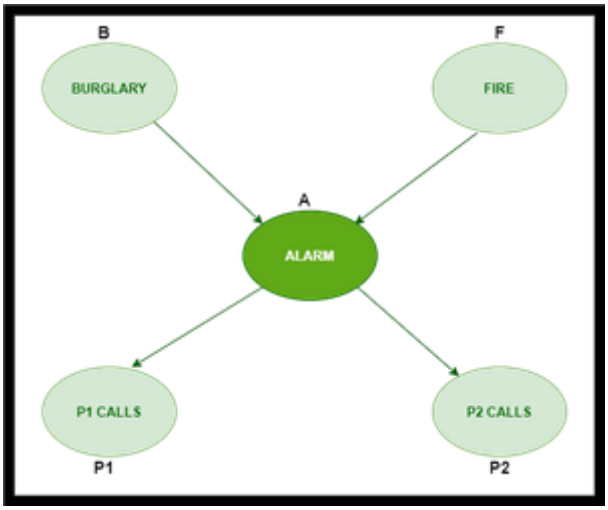
Conditional Probability:

- Conditional probability defines the probability that event B will occur, given that event A has already occurred.

```
If A and B are dependent events  p(B/A) = P( A and B)/P(A)
If A and B are independent events : P(B/A) = P(B)
```

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

Example:



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**

Person 'P1' –

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

Person 'P2' –

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

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

Applications

- **Medical Diagnosis**: Modeling diseases and symptoms to assist in diagnostic reasoning.
- **Machine Learning**: Feature selection, classification, and regression.
- **Natural Language Processing**: Dependency parsing and language modeling.
- **Econometrics**: Understanding relationships between economic indicators.

Markov Random Fields:

- Markov Random Fields (MRFs), also known as Markov Networks, are a type of probabilistic graphical model that represents the dependencies among a set of random variables using an undirected graph.
- They are particularly useful for modeling scenarios where the exact direction of dependency is not well-defined or when symmetrical relationships between variables exist.

- A Markov Random Field or Markov Network is a class of graphical models with an undirected graph between random variables.
- The structure of this graph decides the dependence or independence between the random variables.

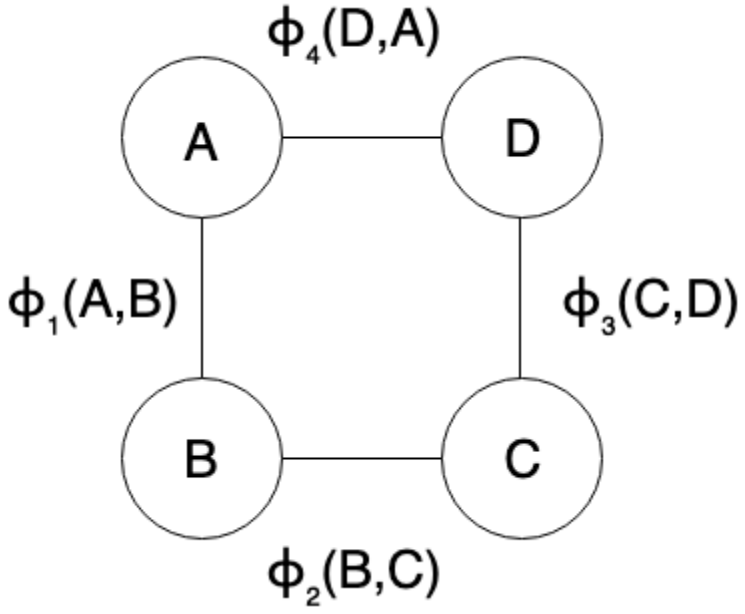


Fig. Markov Random Field with four random variables

Components of Markov Random Fields

1. **Nodes (Vertices)**:
   - Each node represents a random variable.
   - Nodes can represent observed data, hidden variables, or any entities in the model.
2. **Edges (Links)**:
   - Undirected edges between nodes indicate direct dependencies.
   - Unlike Bayesian Networks, MRFs use undirected edges to capture the symmetrical nature of relationships.
3. **Clique Potentials (Factors)**:
   - Potential functions are associated with cliques (fully connected subgraphs) of the graph.
   - They represent the local dependencies among the variables in a clique.
   - These potential functions are often denoted as ψ

Applications

- **Image Processing**: Image segmentation, denoising, and restoration.
- **Natural Language Processing**: Part-of-speech tagging, named entity recognition.
- **Computer Vision**: Object recognition, scene labeling.
- **Bioinformatics**: Modeling spatial dependencies in protein structures.

Hidden Markov Models:

- The hidden Markov Model (HMM) is a statistical model that is used to describe the probabilistic relationship between a sequence of observations and a sequence of hidden states.
- It is often used in situations where the underlying system or process that generates the observations is unknown or hidden, hence it has the name "Hidden Markov Model."
- It is used to predict future observations or classify sequences, based on the underlying hidden process that generates the data.

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

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

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

- The **transition probabilities** describe the probability of transitioning from one hidden state to another.

- The **emission probabilities** describe the probability of observing an output given a hidden state.

Hidden Markov Model Algorithm:

Step 1: Define the state space and observation space

The state space is the set of all possible hidden states, and the observation space is the set of all possible observations.

Step 2: Define the initial state distribution

This is the probability distribution over the initial state.

Step 3: Define the state transition probabilities

These are the probabilities of transitioning from one state to another. This forms the transition matrix, which describes the probability of moving from one state to another.

Step 4: Define the observation likelihoods:

These are the probabilities of generating each observation from each state. This forms the emission matrix, which describes the probability of generating each observation from each state.

Step 5: Train the model

The parameters of the state transition probabilities and the observation likelihoods are estimated using the Baum-Welch algorithm, or the forward-backward algorithm. This is done by iteratively updating the parameters until convergence.

Step 6: Decode the most likely sequence of hidden states

Given the observed data, the Viterbi algorithm is used to compute the most likely sequence of hidden states. This can be used to predict future observations, classify sequences, or detect patterns in sequential data.

Step 7: Evaluate the model

The performance of the HMM can be evaluated using various metrics, such as accuracy, precision, recall, or F1 score.

Tracking Methods:

- Tracking methods in machine learning, often referred to as object tracking, involve techniques used to locate and follow an object's position over time in a sequence of frames or images.
- These methods have applications in various fields, including computer vision, robotics, surveillance, and augmented reality.

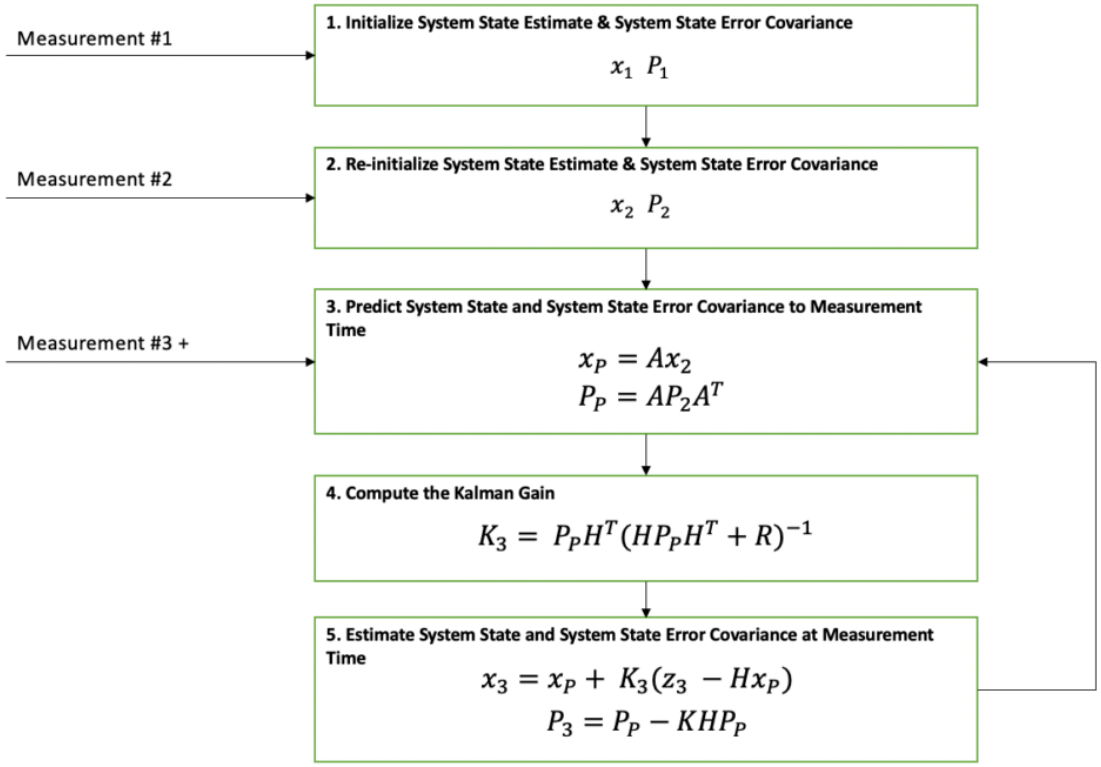
Kalman Filter:

- The Kalman filter is an optimal estimator for linear systems with Gaussian noise.
- It provides a recursive solution to the linear quadratic estimation problem, efficiently processing noisy measurements to produce an estimate of the system's state.

Components:

1. **State Vector (**x_t**)**:
   - Represents the state of the system at time t.
2. **State Transition Model (**A**)**:
   - Describes how the state evolves over time.
   - $x_t=Ax_{t-1}+Bu_t+w_tx_t$ where u_t is the control input and w_t is the process noise.
3. **Measurement Model (**H**)**:
   - Relates the state to the observations.
   - $z_t=Hx_t+v_tz_t$ = where z_t is the measurement and v_t is the measurement noise.
4. **Covariance Matrices (**Q **and** R**)**:
   - Q: Process noise covariance.
   - R: Measurement noise covariance.

Algorithm:



1. **Prediction**:
   - Predict the next state
   - Predict the error covariance
2. **Update**:
   - Compute the Kalman gain
   - Update the state estimate
   - Update the error covariance

Applications:

- **Navigation Systems**: GPS and inertial navigation.
- **Control Systems**: Robotics, aerospace.
- **Economics**: Estimating trends and cycles.

Particle Filter:

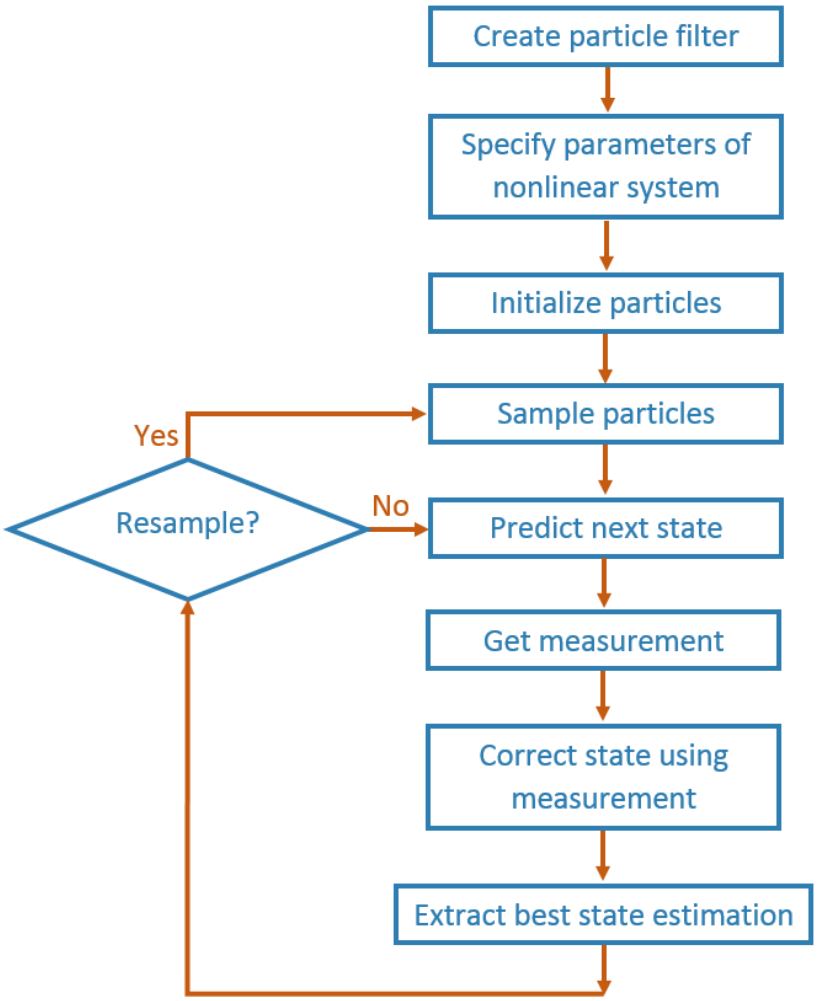
- The particle filter, or Sequential Monte Carlo (SMC) method, is used for non-linear, non-Gaussian systems.
- It represents the posterior distribution of the state using a set of random samples (particles) and weights.

Components

1. **Particles**:
   - A set of samples representing possible states.
2. **Weights**:

- Importance weights for each particle, representing the likelihood given the observations.

Algorithm:



1. **Initialization**:
   - Generate an initial set of particles from the prior distribution.
   - Initialize weights
2. **Prediction**:
   - Propagate particles according to the state transition model
3. **Update**:
   - Update weights based on the measurement likelihood
   - Normalize weights
4. **Resampling**:
   - Resample particles based on their weights to avoid degeneracy.

Applications:

- **Robotics**: Localization and mapping (SLAM).
- **Computer Vision**: Object tracking.
- **Finance**: Filtering in stochastic volatility models.

Comparison:

- **Kalman Filter**:
  - Assumes linear dynamics and Gaussian noise.
  - Computationally efficient.
  - Optimal for linear systems.
- **Particle Filter**:
  - Handles non-linear and non-Gaussian systems.
  - More computationally intensive.
  - Provides a flexible framework for complex systems.

*****