# Principles of Autonomy and Decision Making

**(AI_PrincAutonomy_2808)**

Week 7: Advanced Decision Making Under Uncertainty

Technische Hochschule Ingolstadt


**Team:**

- **Prof. Dr. rer. nat. Lenz Belzner**
- **Chidvilas Karpenahalli Ramakrishna, M.Eng.**
- **Adithya Mohan, M.Eng.**
- **Zahra Zeinaly, Ph.D.**

# Recap

- No Uncertainty
  - Uninformed Search
    - Depth-First Search (DFS) – No consideration
    - Breadth-First Search (BFS) – No consideration
    - Uniform Cost Search (UCS) – Cost consideration
  - Informed Search
    - Greedy Search – Goal information
    - A* algorithm – Cost + Goal information
- Uncertainty in transition (MDPs)
  - Value-Iteration
  - Policy-Iteration
- What other forms of uncertainty can we consider?
  - Uncertainty in the state information
  - Uncertainty in modeling

# Decision Making Under Uncertainty

- Involves choosing actions in environments where
  - Outcomes are not deterministic – MPDs
  - States are not fully observable – POMDPs

- Uncertainty types:
  - Transition Uncertainty
  - State Uncertainty
  - Model Uncertainty

- Importance:
  - Most real-world problems have some degree of uncertainty
  - We need a formulation that can capture these uncertainties
  - Examples:
    - Autonomous Vehicles: Decision making with uncertain sensor data
    - Robotics: Navigation and interation with dynamic environments
    - Healthcare: Diagnosis with uncertainty in test results
    - Finance: Investments with uncertain market conditions

# Challenges

1. Stochastic Environments:
   - Actions may lead to different outcomes, even when performed in the same state
   - Problem formulation using *Markov Decision Process (MDP)*
   - Probabilistic transitions
     - $T(s, a, s') = P(s'|s, a)$
     - Outcomes are described by probabilistic distributions rather than deterministic functions

2. Incomplete Information:
   - Partial observability:
     - Lack of complete knowledge about the current state of the environment
     - Only indirect/ noisy observations of the state are available
   - Problem formulation using *Partially Observable Markov Decision Process (POMDP)*

3. Computational Complexity:
   - Algorithms should handle large state and action spaces under uncertainty
   - Approximation techniques are used
     - Monte Carlo (MC) Methods
     - Particle Filtering

# Partially Observable Markov Decision Process (POMDP)

- Extension / generalization of MDP where the agent does not have full observability of the state

- Components:
  - Set of states ($S$)
  - Set of actions ($A$)
  - Transition function/ probability ($T$)
    - Considers uncertainty in transition
    - $T(s, a, s') = P(s'|s, a)$
    - Probability of transitioning from state $s$ to $s'$ when choosing an action $a$
  - Reward function
    - $R(s, a, s')$ or $R(s, a)$
  - Set of observations ($\Omega$)
  - Observation model/ probability ($O$)
    - Considers uncertainty in the current state
    - $O(o, s', a) = P(o|s', a)$ is the probability of observing $o$ after taking action $a$ and transitioning into $s'$
    - Here, $o \in \Omega$
- Goal:
  - Maximum expected cumulative reward over time

# POMDP: Belief State and Observation

- Belief state:
  - It is a probability distribution over all possible states, representing the agent‘s uncertainty about the current state of the system
  - Denoted as $b(s)$
  - $b(s)$ denotes the probability (belief) that the system is in state $s$

- Observation:
  - It is information received by the agent that provides indirect evidence about the actual state of the system

- Observation Model:
  - Gives the probability of receiving observation $o$ given the agent is in the state $s'$ after taking action $a$
  - Denoted as $P(o|s', a)$

- NOTE:
  - Observations help update the belief state to better reflect the current state of the agent

# POMDP: Bellman Equation

- *MDP* Bellman equation:

$$V(s) = max_{a \in A}\left[\sum_{s' \in S} P(s'|s,a)\left(R(s,a,s') + \gamma V(s')\right)\right]$$

- *POMDP* Bellman equation:

$$V(b) = max_{a \in A}\left[\sum_{s \in S} b(s)\left\{\sum_{s' \in S} P(s'|s,a)\left\langle R(s,a,s') + \gamma \sum_{o \in O} P(o|s',a)V(b')\right\rangle\right\}\right]$$

$$\pi^*(b) = argmax_{a \in A}\left[\sum_{s \in S} b(s)\left\{\sum_{s' \in S} P(s'|s,a)\left\langle R(s,a,s') + \gamma \sum_{o \in O} P(o|s',a)V(b')\right\rangle\right\}\right]$$

Here,

- Red: Captures uncertainty in the current state
- Orange: Captures uncertainty in the transition
- Green: Captures uncertainty in the resulting state

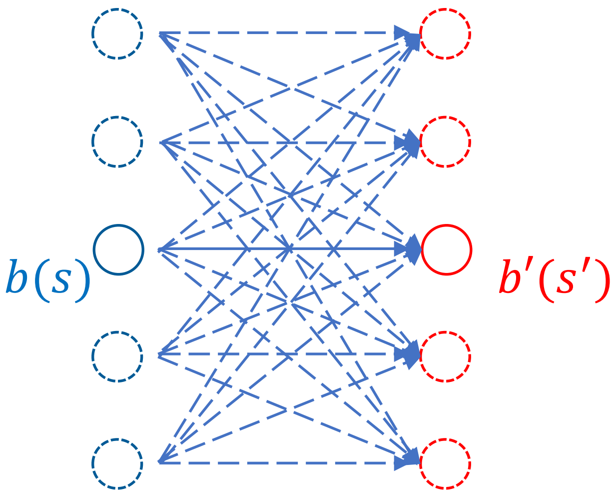


Fig 1. Depiction of uncertainty in states

# POMDP: Belief Update

- We update our belief of a state after taking an action $a$ and receiving an observation $o$
- *Bayes'* rule for belief update:

$$b'(s') = \eta P(o|s', a) \sum_{s \in S} P(s'|s, a) b(s)$$

  - Here,
    - $b(s)$ is the probability that the system is in state $s$ before taking any action given belief state $b$
    - $b'(s')$ is the probability that the system is in state $s'$ after taking an action $a$ and receiving an observation $o$ given belief state $b'$
    - $\eta$ is a normalizing factor
    - $P(o|s', a)$ is the observation probability
    - $P(s'|s, a)$ is the transition probability

# POMDP: Steps for Belief Update

1. Prediction step:
   - For „each possible“ new state $s'$, predict the new belief state using the transition model
   - Calculate intermediate belief $\hat{b}(s')$ by summing over “all possible” previous states $s$

$$\hat{b}(s') = \sum_{s \in S} P(s'|s,a)b(s)$$

2. Calculate normalization factor:
   - $\eta$ is used to ensure that the belief is a valid probability distribution (i.e. it sums to 1)

$$\eta = \frac{1}{\sum_{s'} P(o|s',a)\hat{b}(s')} = \frac{1}{\sum_{s'} P(o|s',a)\sum_{s \in S} P(s'|s,a)b(s)}$$

3. Correction step:
   - Correct the intermediate belief based on the observation received

$$b'(s') = \eta P(o|s',a)\hat{b}(s')$$

$$b'(s') = \eta P(o|s',a)\sum_{s \in S} P(s'|s,a)b(s)$$

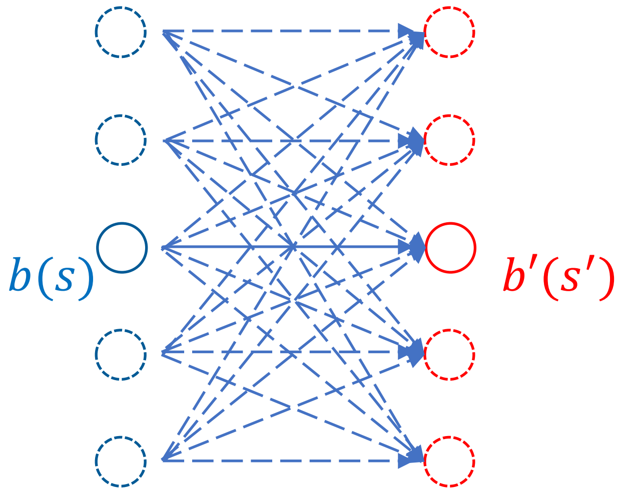


Fig 1. Depiction of uncertainty in states

# Recap: Bayes‘ Theorem

- Bayes‘ Theorem is used to update the probability of a hypothesis based on new evidence

$$P(H|E) = \frac{P(E|H).P(H)}{P(E)}$$

- Here,
  - $P(H|E)$ is the posterior probability of the hypothesis $H$ given the evidence $E$
  - $P(E|H)$ is the likelihood of the evidence $E$ given that the hypothesis $H$ is true
  - $P(H)$ is the prior probability of the hypothesis $H$
  - $P(E)$ is the marginal probability of the evidence $E$

# Bayesian Decision Making

- Bayesian Belief Update in POMDPs:

$$b'(s') = \frac{P(o|s', a) \sum_{s \in S} P(s'|s, a) b(s)}{\sum_{s'} P(o|s', a) \sum_{s \in S} P(s'|s, a) b(s)}$$

  - Here,
    - $b'(s')$: Captures the probability of the hypothesis that we are in state $s'$ given the evidence/observation $o$
    - $P(o|s', a)$: Captures the probability of the evidence $o$ given the hypothesis $s'$ and $a$
    - $\sum_{s \in S} P(s'|s, a) b(s)$: Captures the probability of the hypothesis
    - $\sum_{s'} P(o|s', a) \sum_{s \in S} P(s'|s, a) b(s)$: Captures the probability of the evidence/observation

- Example: Robot Navigation
  1. Initial belief:
     - Robot starts with a belief state $b(s)$ representing its uncertain position
  2. Action taken:
     - The robot moves forward by taking action $a$ and receives a sensor reading $o$
  3. Belief Update:
     - Using Bayesian Inference, the robot updates its belief state to b$'(s')$

- Summary:
  - Bayesian decision-making in POMDPs uses belief states to manage uncertainty
  - Belief states are updated through Bayesian inference after each action and observation
  - The value function and optimal policy are computed using the updated beliefs

# Monte Carlo Methods

- MC methods are a class of algorithms that rely on „random sampling“ to obtain numerical results

- Necessity:
  - The belief space is high dimensional and very complex to model

- MC methods:
  1. Monte Carlo Sampling for Belief States
     - Used to approximate the belief state distribution
     - Generates a large number of random samples (state-action-observation) from the belief state
     - Uses these samples to estimate the probability distribution of future states and observations

  2. Monte Carlo Integration
     - Used to evaluate expected rewards and value functions

  3. Monte Carlo Tree Search (MCTS)
     - Used to find the optimal policy by exploring the most promising actions

# Particle Filtering (Sequential Monte Carlo Method)

- Particle Filtering is a specialized form of MC method specifically designed for sequential state estimation in uncertain environments

- Particle:
  - Samples from the belief state distribution

- Steps:
  1. Initialization:
     - Initialize $N$ particles $\{s_1, s_2, \ldots, s_N\}$ according to the prior belief distribution $b_0(s)$
  2. Prediction (sampling):
     - For each particle $s_i$, sample a new state $s'_i$ based on the transition model $P(s_i'|s_i, a)$
  3. Update (Weighting):
     - For each predicted particle $s'_i$, compute the weight $w_i$ based on the observation model $P(o|s'_i, a)$
       $$w_i = P(o|s'_i, a)$$
  4. Normalization:
     - Normalize the weights $w_i$ by dividing each weight by the sum of all weights
  5. Resampling:
     - Focus on the most probable particles by resampling based on the weights
     - Resample $N$ particles from the current set $\{(s'_i, w_i)\}$ with replacement, where the probability of selecting each particle is proportional to its weight
  6. Iteration:
     - Repeat the prediction, update, normalization, and resampling steps at each time step as new actions are taken and observations are received

# Particle Filtering: Example

- Problem: Robot Localization Using Particle Filtering

- Scenario:
  - A robot navigates a building, updating its belief about its location using particle filtering

- Steps:
  - Initialization:
    - Start with particles uniformly distributed across possible locations
  - Prediction:
    - For each particle, predict the new location after the robot moves
  - Update:
    - Compute weights based on the likelihood of the robot's sensor readings
  - Normalization:
    - Normalize the weights to ensure they sum to 1
  - Resampling:
    - Resample particles to focus on the most likely locations
  - Iteration:
    - Continue the process as the robot moves and receives new sensor readings