Reasoning under uncertainty

Universidad Carlos III de Madrid

AI

Outline

1. Introduction
2. Probabilistic reasoning
3. Bayesian networks
4. Markov Models
5. Fuzzy logic

Outline

1. Introduction
2. Probabilistic reasoning
3. Bayesian networks
4. Markov Models
5. Fuzzy logic

Outline

1. Introduction
2. Probabilistic reasoning
3. Bayesian networks
4. Markov Models
5. Fuzzy logic

Outline

1. Introduction
2. Probabilistic reasoning
3. Bayesian networks
4. Markov Models
5. Fuzzy logic

Outline

1 Introduction

2 Probabilistic reasoning

3 Bayesian networks

4 Markov Models
   - Introduction
   - Markov chains
   - Hidden Markov Models
   - Markov Decision Processes (MDP)
   - Partially Observable MDPs (POMDPs)

5 Fuzzy logic

Outline

1. Introduction
2. Probabilistic reasoning
3. Bayesian networks
4. Markov Models
   - Introduction
   - Markov chains
   - Hidden Markov Models
   - Markov Decision Processes (MDP)
   - Partially Observable MDPs (POMDPs)
5. Fuzzy logic

Summary of Probability and BN

- Probabilities can represent uncertainty in AI
- Inference by enumeration solves questions such as P(event|observation)
- Conditional independence allows for a more compact representation of probability distribution
- Bayesian networks is a graph representation based on conditional independence
- We can use exact or stochastic methods for inference

Motivation: Probability + time + actions

- Probabilistic reasoning:
  - How can we add the notion of time?
    - what is the probability of a user cliking in one of our web pages if she is now visiting one of our web pages?
    - Idea: We can create random variables for each time step: $\text{Web}_t, \text{Web}_{t+1}, \ldots$

Motivation: Probability + time + actions

- Probabilistic reasoning:
  - How can we add the notion of time?
    - what is the probability of a user cliking in one of our web pages if she is now visiting one of our web pages?
    - Idea: We can create random variables for each time step: $\text{Web}_t, \text{Web}_{t+1}, \ldots$
  - How can we use probabilities to select actions to solve a problem?

Motivation: Probability + time + actions

- Probabilistic reasoning:
  - How can we add the notion of time?
    - what is the probability of a user cliking in one of our web pages if she is now visiting one of our web pages?
    - Idea: We can create random variables for each time step: $\text{Web}_t, \text{Web}_{t+1}, \ldots$
  - How can we use probabilities to select actions to solve a problem?
    - we have used search algorithms to select a sequence of actions to solve a problem, but what happens when actions are not deterministic?
    - how can we select actions such that we have the highest probability of reaching the goal state?

Motivation: Probability + time + actions

- Probabilistic reasoning:
  - How can we add the notion of time?
    - what is the probability of a user cliking in one of our web pages if she is now visiting one of our web pages?
    - Idea: We can create random variables for each time step: $\text{Web}_t, \text{Web}_{t+1}, \ldots$
  - How can we use probabilities to select actions to solve a problem?
    - we have used search algorithms to select a sequence of actions to solve a problem, but what happens when actions are not deterministic?
    - how can we select actions such that we have the highest probability of reaching the goal state?
- Tasks
  - Markov chains
  - Hidden Markov Models (HMMs)
  - Markov Decision Processes (MDPs)
  - Partially Observable MDPs (POMDPs)

Outline

1. Introduction
2. Probabilistic reasoning
3. Bayesian networks
4. Markov Models
   - Introduction
   - Markov chains
   - Hidden Markov Models
   - Markov Decision Processes (MDP)
   - Partially Observable MDPs (POMDPs)
5. Fuzzy logic

Markov Chains (Markov Processes)

- A sequence of random variables $X_0, X_1, \ldots, X_t$
- They represent the value of a random variable X over time
- Full observability
- Can be modeled as a chain-structured Bayesian network

$$X_0 \rightarrow X_1 \rightarrow X_2 \rightarrow \ldots$$

  - each node is identically distributed
  - (belief) state: probability distribution of X at a given time
  - initial state (probability distribution): $P(X_0)$
  - transition probabilities: how state evolves over time

$$P(X_t | X_{t-1})$$

Markov Chains (Markov Processes)

- A sequence of random variables $X_0, X_1, \ldots, X_t$
- They represent the value of a random variable X over time
- Full observability
- Can be modeled as a chain-structured Bayesian network

$$X_0 \to X_1 \to X_2 \to \ldots$$

  - each node is identically distributed
  - (belief) state: probability distribution of X at a given time
  - initial state (probability distribution): $P(X_0)$
  - transition probabilities: how state evolves over time

$$P(X_t | X_{t-1})$$

- Markov property: the past and the future are independent given the present

Example. User visiting our/other web pages

- States:
  - $X_0 = X_1 = \ldots =$ {other, our}

Example. User visiting our/other web pages

- States:
  - $X_0 = X_1 = \ldots =$ {other, our}
  - $X_{t-1} = X_t =$ {other, our}

Example. User visiting our/other web pages

- States:
  - $X_0 = X_1 = \ldots =$ {other, our}
  - $X_{t-1} = X_t =$ {other, our}
- Transitions:

X_{t-1}	$P(X_t$=our)	$P(X_t$=other)
our	0.9	0.1
other	0.1	0.9

Example. User visiting our/other web pages

- States:
  - $X_0 = X_1 = \ldots =$ {other, our}
  - $X_{t-1} = X_t =$ {other, our}
- Transitions:

X_{t-1}	$P(X_t$=our)	$P(X_t$=other)
our	0.9	0.1
other	0.1	0.9

- Initial state: $P(X_0 =$ our$) = 1.0$, $P(X_0 =$ other$) = 0.0$

Example. User visiting our/other web pages

- States:
  - $X_0 = X_1 = \ldots =$ {other, our}
  - $X_{t-1} = X_t =$ {other, our}
- Transitions:

X_{t-1}	$P(X_t\text{=our})$	$P(X_t\text{=other})$
our	0.9	0.1
other	0.1	0.9

- Initial state: $P(X_0 = \text{our}) = 1.0$, $P(X_0 = \text{other}) = 0.0$
- What is the probability distribution after one step?

Example. User visiting our/other web pages

- States:
  - $X_0 = X_1 = \ldots =$\{other, our\}
  - $X_{t-1} = X_t =$\{other, our\}
- Transitions:

X_{t-1}	$P(X_t{=}\text{our})$	$P(X_t{=}\text{other})$
our	0.9	0.1
other	0.1	0.9

- Initial state: $P(X_0 = \text{our}) = 1.0$, $P(X_0 = \text{other}) = 0.0$
- What is the probability distribution after one step?

$$
\begin{aligned}
P(X_1 = \text{our}) = \textstyle\sum_{i \in \{\text{our,other}\}} & P(X_1 = \text{our} | X_0 = i) P(X_0 = i) = \\
& P(X_1 = \text{our} \mid X_0 = \text{our}) P(X_0 = \text{our}) + \\
& P(X_1 = \text{our} \mid X_0 = \text{other}) P(X_0 = \text{other}) = \\
& 0.9 \times 1.0 + 0.1 \times 0.0 = 0.9
\end{aligned}
$$

$$P(X_1 = \text{other}) = 1 - P(X_1 = \text{our}) = 0.1$$

How about predicting the future?

- From initial observation of our

$$\begin{pmatrix} 1.0 \\ 0.0 \end{pmatrix} \quad \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix} \quad \begin{pmatrix} 0.82 \\ 0.18 \end{pmatrix} \quad \cdots \quad \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

$$P(X_0) \quad P(X_1) \quad P(X_2) \quad \ldots \quad P(X_\infty)$$

- From initial observation of other

$$\begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix} \quad \begin{pmatrix} 0.1 \\ 0.9 \end{pmatrix} \quad \begin{pmatrix} 0.18 \\ 0.82 \end{pmatrix} \quad \cdots \quad \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

$$P(X_0) \quad P(X_1) \quad P(X_2) \quad \ldots \quad P(X_\infty)$$

Stationary Distributions

- If we simulate the chain long enough
  - uncertainty accumulates
  - eventually, we have no idea what the state is!
- Stationary distributions
  - for most chains, the distribution we end up in is independent of the initial distribution
  - usually, we can only predict a short time out

Outline

1 Introduction

2 Probabilistic reasoning

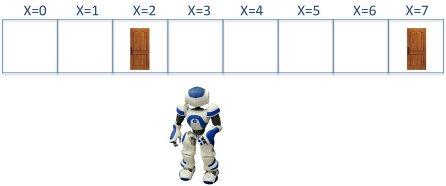
3 Bayesian networks

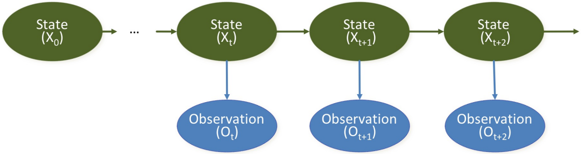
4 Markov Models
- Introduction
- Markov chains
- Hidden Markov Models
- Markov Decision Processes (MDP)
- Partially Observable MDPs (POMDPs)

5 Fuzzy logic

Hidden Markov Models

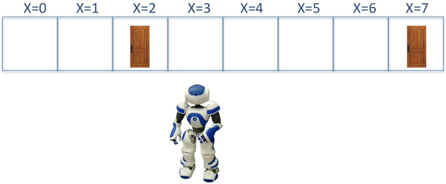
- Observations (partial)
- As a BN (localize a robot):

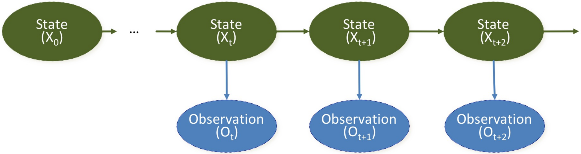




Hidden Markov Models

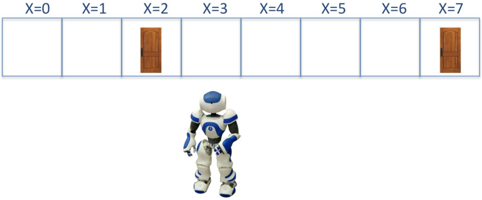
- Observations (partial)
- As a BN (localize a robot):

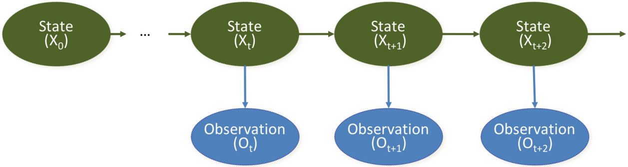




- Independence properties:
  - Markov property (influences the values of X_t)
  - the current observation is independent of everything else given current state (influences the values of O_t)

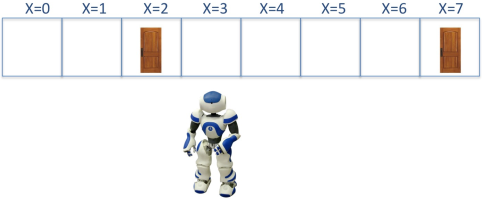
Example of HMM

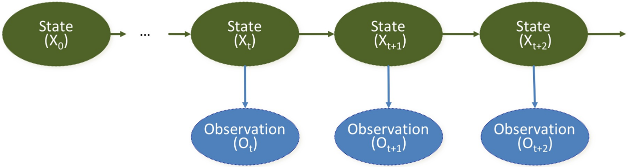




- State: probability distribution of the position of robot $P(X_t)$
- Initial state: $X_0 = 3 \rightsquigarrow$
  $P(X_0 = 0) = 0.0 \ldots P(X_0 = 3) = 1.0 \ldots P(X_0 = 7) = 0.0$
- Observation: it sees a door or not $P(Door)$

Simple example of HMM

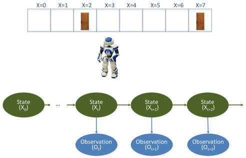




An HMM is defined by:

- Initial distribution: $P(X_0)$
- Transitions: $P(X_t|X_{t-1})$
- Observations (emissions): $P(O_t|X_t) = P(O|X)$

Ejemplo de HMM



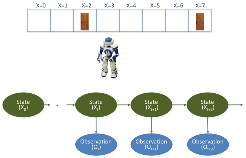
- Distribución inicial: $P(X_0) = \langle 0, 0, 0, 1, 0, 0, 0, 0 \rangle$
- Transiciones: $P(X_t | X_{t-1})$

X_{t-1}	$P(X_t = 0)$	$P(X_t = 1)$	$\dots$	$P(X_t = 7)$
0	0.1	0.9	$\dots$	0.0
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
7	0.0	0.0	$\dots$	0.1

- Observaciones: $P(O_t | X_t) = P(O | X)$

X_t	$P(\text{Puerta}_t = yes)$	$P(\text{Puerta}_t = no)$
0	0.0	1.0
$\dots$	$\dots$	$\dots$
7	1.0	0.0

Simple example of HMM



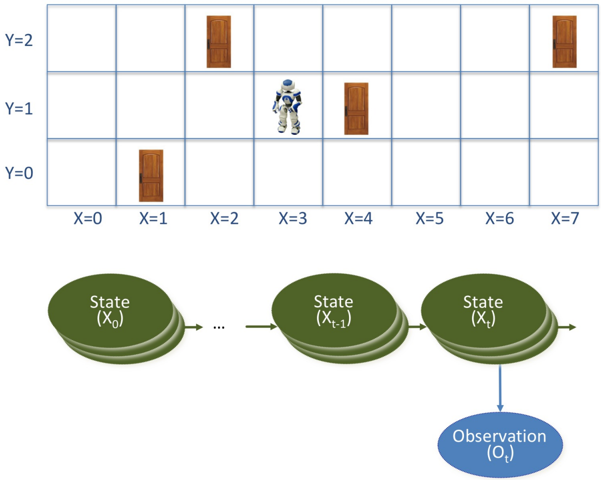
- Initial distribution: $P(X_0) = \langle 0, 0, 0, 1, 0, 0, 0, 0 \rangle$
- Transitions: $P(X_t | X_{t-1})$

X_{t-1}	$P(X_t = 0)$	$P(X_t = 1)$	$\ldots$	$P(X_t = 7)$
0	0.1	0.9	$\ldots$	0.0
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
7	0.0	0.0	$\ldots$	0.1

- Observations: $P(O_t | X_t) = P(O | X)$

X_t	$P(\mathrm{Door}_t = yes)$	$P(\mathrm{Door}_t = no)$
0	0.0	1.0
$\ldots$	$\ldots$	$\ldots$
7	1.0	0.0

More complex HMM



More complex HMM

- States: $S = (X, Y)$
- Probability distributions of states:
  - $P(X) = \langle P(X = 0), P(X = 1), \ldots, P(X = 7) \rangle$
  - $P(Y) = \langle P(Y = 0), P(Y = 1), P(Y = 2) \rangle$
  - $P(S) = P(X, Y) = \langle P(X = 0, Y = 0), \ldots, P(X = 7, Y = 2) \rangle$
- Initial distribution:
  - $P(X_0) = \langle 0, 0, 0, 1, 0, 0, 0, 0 \rangle$
  - $P(Y_0) = \langle 0, 1, 0 \rangle$
  - $P(S_0) = \langle 0, 0, \ldots, 1, \ldots, 0 \rangle$
- Observations: robot sees door (yes, no)

More complex HMM

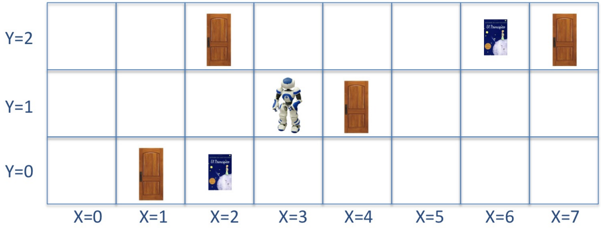
- Transitions: $P(S_t|S_{t-1})$

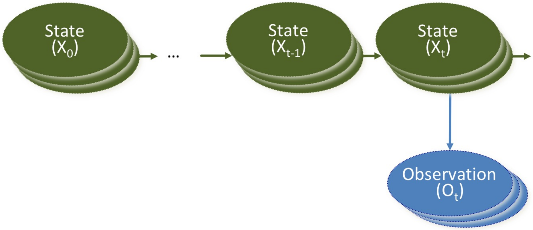
X_{t-1}	Y_{t-1}	$P(X_t = 0)$	$\dots$	$P(X_t = 7)$	$P(Y_t = 0)$	$P(Y_t = 1)$	$P(Y_t = 2)$
0	0	0.3	$\dots$	0.0	0.4	0.6	0.0
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	
7	2	0.0	$\dots$	0.6	0.0	0.4	0.6

- Observations: $P(O_t|S_t) = P(O|S) = P(O|X, Y)$

X_t	Y_t	$P(O_t = yes)$	$P(O_t = no)$
0	0	0.0	0.0
1	0	1.0	0.0
$\dots$	$\dots$	$\dots$	$\dots$
7	2	1.0	0.0

Yet a more complex HMM

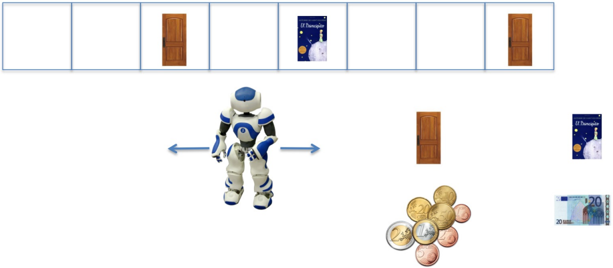




Outline

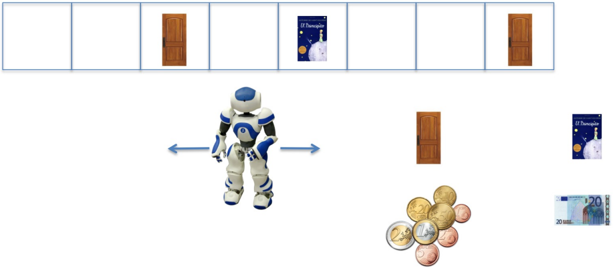
1. Introduction
2. Probabilistic reasoning
3. Bayesian networks
4. Markov Models
   - Introduction
   - Markov chains
   - Hidden Markov Models
   - Markov Decision Processes (MDP)
   - Partially Observable MDPs (POMDPs)
5. Fuzzy logic

Markov Decision Processes (MDP)



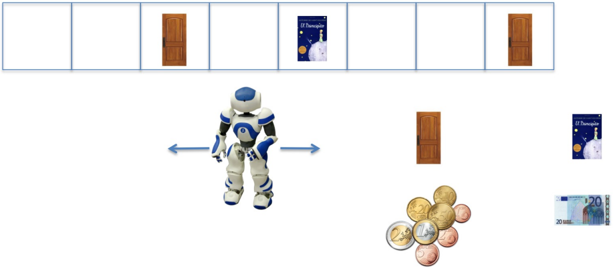
- In HMMs we were following the robot: inferring where it was (from partial observations)
- In MDP, we want to control the robot:

Markov Decision Processes (MDP)



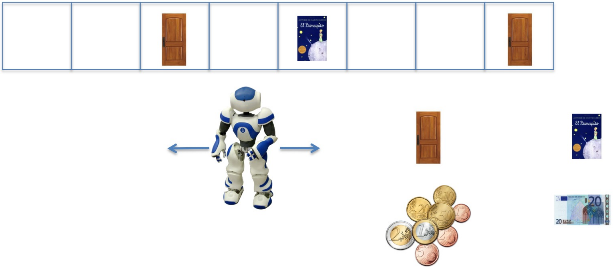
- In HMMs we were following the robot: inferring where it was (from partial observations)
- In MDP, we want to control the robot:
  - select actions

Markov Decision Processes (MDP)



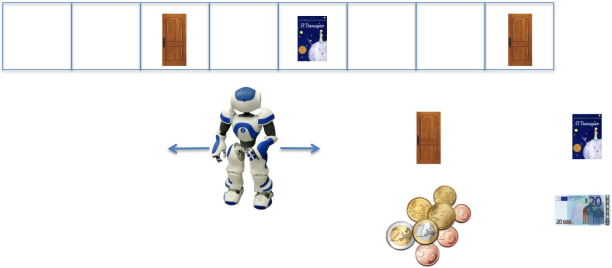
- In HMMs we were following the robot: inferring where it was (from partial observations)
- In MDP, we want to control the robot:
  - select actions
  - such that they maximize the reward

Markov Decision Processes (MDP)



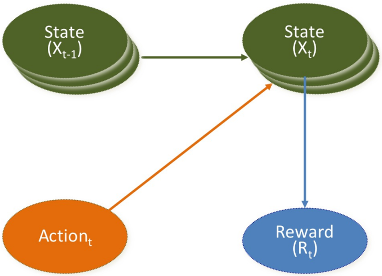
- In HMMs we were following the robot: inferring where it was (from partial observations)
- In MDP, we want to control the robot:
  - select actions
  - such that they maximize the reward
  - over time

Markov Decision Processes (MDP)



- In HMMs we were following the robot: inferring where it was (from partial observations)
- In MDP, we want to control the robot:
  - select actions
  - such that they maximize the reward
  - over time
  - based on full observations: complete state

Markov Decision Processes (MDP). BN's view



Markov Decision Process (MDP)

- Markov process where we make a decision at every round
- MDP: $< S, A, T, R >$
  - S: set of states (including initial and goal state)
  - A: set of actions
  - T: transition function

$$T(s, a, s') = P(S_{t+1} = s' | S_t = s, A_t = a)$$

  - R: reward function

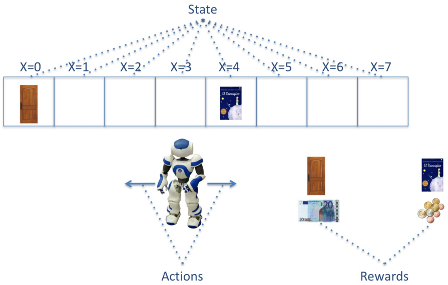
$$R(S_t = s, A_t = a) = R(s, a)$$

    - sometimes just $R(s)$

Example

MDP Tuple: $< S, A, T, R >$

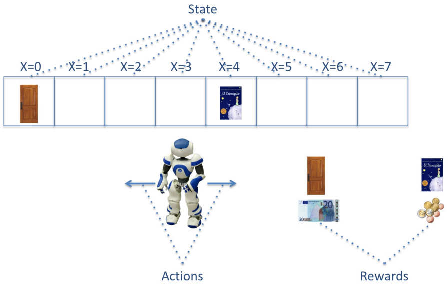
- S: position of robot
  - initial state: $X_0 = 3$
  - goal state: $X_n = 0$



Example

MDP Tuple: $< S, A, T, R >$

- S: position of robot
  - initial state: $X_0 = 3$
  - goal state: $X_n = 0$
- A: left, right

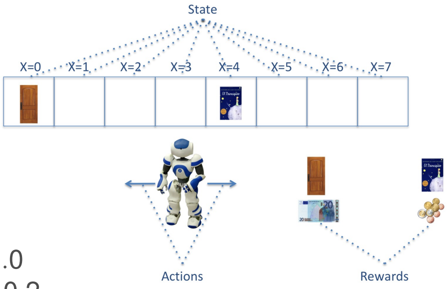


Example

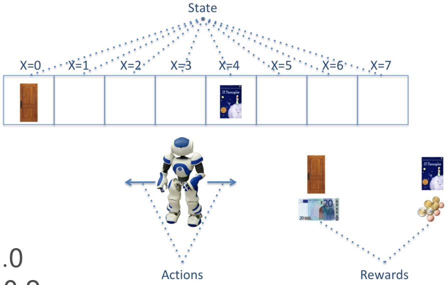
MDP Tuple: $< S, A, T, R >$

- S: position of robot
  - initial state: $X_0 = 3$
  - goal state: $X_n = 0$
- A: left, right
- T: transition function
  - $P(X_t = 0 | X_{t-1} = 0, \text{left}) = 1.0$
  - $P(X_t = 0 | X_{t-1} = 0, \text{right}) = 0.2$
  - $P(X_t = 1 | X_{t-1} = 0, \text{left}) = 0.0$
  - $P(X_t = 1 | X_{t-1} = 0, \text{right}) = 0.8$
  - $\ldots$

  If action is not applicable, the robot stays in the same position



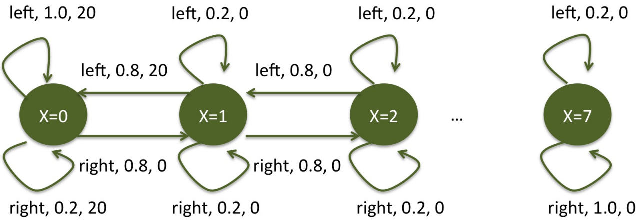
Example

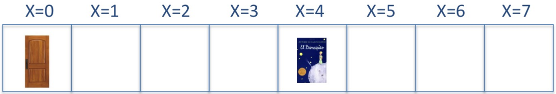


MDP Tuple: $< S, A, T, R >$

- S: position of robot
  - initial state: $X_0 = 3$
  - goal state: $X_n = 0$
- A: left, right
- T: transition function
  - $P(X_t = 0 | X_{t-1} = 0, \text{left}) = 1.0$
  - $P(X_t = 0 | X_{t-1} = 0, \text{right}) = 0.2$
  - $P(X_t = 1 | X_{t-1} = 0, \text{left}) = 0.0$
  - $P(X_t = 1 | X_{t-1} = 0, \text{right}) = 0.8$
  - $\ldots$
    If action is not applicable, the robot stays in the same position
- R: reward
  - $R(X = 0, \text{left}) = R(X = 0, \text{right}) = R(X = 0) = 20$
  - $R(X = 1, \text{left}) = R(X = 1, \text{right}) = R(X = 1) = 0$
  - $\ldots$
  - $R(X = 4, \text{left}) = R(X = 4, \text{right}) = R(X = 4) = 3.88$
  - $\ldots$

Example. MDP as a Probabilistic state machine





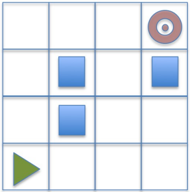
Markov Decision Process (MDP)

- MDP: $< S, A, T, R >$
- Task: choose a sequence of actions
    - not just one decision or one action
    - utility based on a sequence of decisions

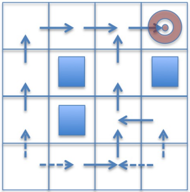
Markov Decision Process (MDP)

- MDP: $< S, A, T, R >$
- Task: choose a sequence of actions
  - not just one decision or one action
  - utility based on a sequence of decisions
- In order to choose a sequence of actions, a policy is computed

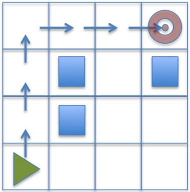
Path (plan) vs Policy (Deterministic)



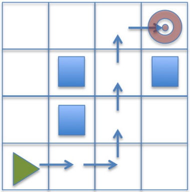
Task



Optimal policy



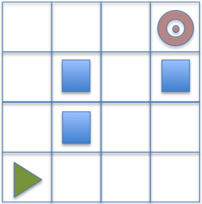
Optimal path



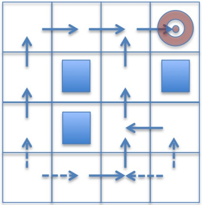
Optimal path

Path (plan) vs Policy (probabilistic)

- Stochastic Action: achieves the desired effect with 0.8 probability
- Example: Action $\leftarrow$ moves to the Left with 0.8 probability, Up with 0.1 probability and Down with 0.1 probability
- There is no optimal path
- Policy knows what to do regardless of the effect of any action at any time-step



Task



Optimal policy

Policies

- Policy: complete mapping from states to actions
- Policy is like a plan, but not quite
  - generated ahead of time, like a plan
- Unlike traditional plans, it is not a sequence of actions that an agent must execute
  - if there are failures in execution, the agent can continue executing a policy
  - prescribes an action for all states

Policies

- Policy: complete mapping from states to actions
- Policy is like a plan, but not quite
  - generated ahead of time, like a plan
- Unlike traditional plans, it is not a sequence of actions that an agent must execute
  - if there are failures in execution, the agent can continue executing a policy
  - prescribes an action for all states
- Maximizes expected reward, rather than just reaching a goal state

Policies

- Policy: complete mapping from states to actions
- Policy is like a plan, but not quite
  - generated ahead of time, like a plan
- Unlike traditional plans, it is not a sequence of actions that an agent must execute
  - if there are failures in execution, the agent can continue executing a policy
  - prescribes an action for all states
- Maximizes expected reward, rather than just reaching a goal state
- For every MDP there exists an optimal policy
  - for every possible start state, there is no better option than to follow the policy

Optimal policy

- We can reason about
  - maximizing expected reward, or
  - minimizing expected cost
- Optimal Policy π^*: lowest expected cost
- Cost models
  - deterministic: if action a always changes the state to s', the cost is

$$c(a) + \mathrm{costFrom}(s')$$

Optimal policy

- We can reason about
  - maximizing expected reward, or
  - minimizing expected cost
- Optimal Policy π^*: lowest expected cost
- Cost models
  - deterministic: if action a always changes the state to s', the cost is

$$c(a) + \mathrm{costFrom}(s')$$

  - stochastic: if action a has a probabilistic effect, the expected cost is (remember the expected utility formula)

$$c(a) + \sum_{s'} P(s'|s, a) \times \mathrm{costFrom}(s')$$

Optimal policy

- We can reason about
  - maximizing expected reward, or
  - minimizing expected cost
- Optimal Policy π^*: lowest expected cost
- Cost models
  - deterministic: if action a always changes the state to s', the cost is

$$c(a) + \mathrm{costFrom}(s')$$

  - stochastic: if action a has a probabilistic effect, the expected cost is (remember the expected utility formula)

$$c(a) + \sum_{s'} P(s'|s, a) \times \mathrm{costFrom}(s')$$

How can we define an optimal policy?

Expected value of a state

V(*s*): expected cost to arrive to the goal from *s*

Expected value of a state

V(*s*): expected cost to arrive to the goal from s

- Search:
  - $V(s)$: cost of optimal path from s
  - If we know $V(s)$, we could use $f(s) = V(s)$ with hill-climbing
  - $V(s)$ is, in fact, a perfect heuristic function $h^*(s)$

Expected value of a state

V(*s*): expected cost to arrive to the goal from *s*

- Search:
  - $V(s)$: cost of optimal path from s
  - If we know $V(s)$, we could use $f(s) = V(s)$ with hill-climbing
  - $V(s)$ is, in fact, a perfect heuristic function $h^*(s)$
- MDP:
  - $V(s)$: expected cost of the optimal strategy to arrive to the goal from s
  - If we know $V(s)$, we can compute optimal policy π^*

Expected value of a state

V(s): expected cost to arrive to the goal from s

- Search:
  - $V(s)$: cost of optimal path from s
  - If we know $V(s)$, we could use $f(s) = V(s)$ with hill-climbing
  - $V(s)$ is, in fact, a perfect heuristic function $h^*(s)$
- MDP:
  - $V(s)$: expected cost of the optimal strategy to arrive to the goal from s
  - If we know $V(s)$, we can compute optimal policy π^*

How do we compute $V(S)$?

Computing $V(S)$. Deterministic

- Bellman Equation:
  - If s is a goal, $V(s) = 0$
  - Otherwise (s' is the new state after applying a at state s):

$$V(s) = \min_{a \in A(s)} [c(a) + V(s')]$$

$$\begin{aligned}
V(S_0) &= \min_{a \in A(s_0)} [c(a) + V(s')] \\
V(S_1) &= \min_{a \in A(s_1)} [c(a) + V(s')] \\
\ldots & \\
V(S_n) &= \min_{a \in A(s_n)} [c(a) + V(s')]
\end{aligned}$$

Computing $V(S)$. Deterministic

- Bellman Equation:
  - If s is a goal, $V(s) = 0$
  - Otherwise (s' is the new state after applying a at state s):

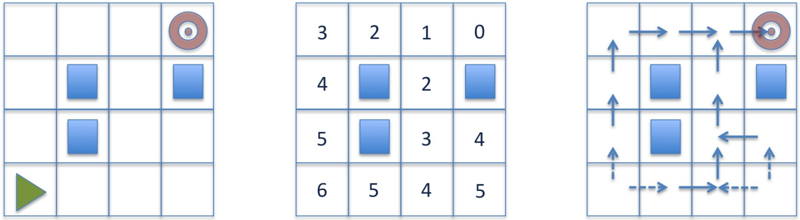
$$V(s) = \min_{a \in A(s)} [c(a) + V(s')]$$

$$\begin{aligned}
V(S_0) &= \min_{a \in A(s_0)} [c(a) + V(s')] \\
V(S_1) &= \min_{a \in A(s_1)} [c(a) + V(s')] \\
\ldots \\
V(S_n) &= \min_{a \in A(s_n)} [c(a) + V(s')]
\end{aligned}$$

- Optimal Policy (s' is the new state after applying a at state s)

$$\pi^*(s) = \arg\min_a [c(a) + V(s')]$$

Path vs Policy (deterministic)



Task Values Optimal policy

Bellman Equation for MDPs

- Stochastic domains
  - the expected value of action *a* (and perfect execution from there):

$$c(a) + \sum_{s' \in S} P(s'|s, a)V(s')$$

Bellman Equation for MDPs

- Stochastic domains
  - the expected value of action a (and perfect execution from there):

$$c(a) + \sum_{s' \in S} P(s'|s,a)V(s')$$

- Bellman equations
  - if s is a goal state, $V(s) = 0$
  - otherwise

$$V(s) = \min_{a \in A(s)} [c(a) + \sum_{s' \in S} P(s'|s,a)V(s')]$$

Bellman Equation for MDPs

- Stochastic domains
  - the expected value of action a (and perfect execution from there):

$$c(a) + \sum_{s' \in S} P(s'|s,a)V(s')$$

- Bellman equations
  - if s is a goal state, $V(s) = 0$
  - otherwise

$$V(s) = \min_{a \in A(s)} [c(a) + \sum_{s' \in S} P(s'|s,a)V(s')]$$

- Optimal Policy

$$\pi^*(s) = \arg\min_a [c(a) + \sum_{s' \in S} P(s'|s,a)V(s')]$$

Path vs Policy (probabilistic)

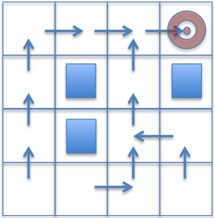
- Stochastic Action: achieves the desired effect with 0.8 probability
- Example: Action $\leftarrow$ moves to the Left with 0.8 probability, Up with 0.1 probability and Down with 0.1 probability

3	2	1	0
4		2	
5		3	4
6	5	4	5

Deterministic values

3.7	2.5	1.2	0.0
5.0		2.5	
6.2		3.9	5.3
7.4	6.7	5.4	6.4

Probabilistic values



Optimal policy

Solving the Bellman equation

Value Iteration Algorithm

$V(s) = 0$ for all states s
while not(Termination) do
    for each state $s \in S$
        $V(s) := \min_{a \in A(s)}[c(a) + \sum_{s' \in S} P(s'|s,a)V(s')]$
Return $[V(s_1), \ldots, V(s_n)]$

Example of Value Iteration

- States: s_1, s_2, s_3
- Goal: s_3
- Action: Right
  - transits from s_i to s_{i+1} with probability 0.8
  - with probability 0.2, it stays in s_i
  - cost: 1
- At start: $V(s_1) = V(s_2) = 0$
- $V(s_3) = 0$ always, because it is the goal

Example of Value Iteration

$$
\begin{aligned}
\text{Bellman eq: } V(s_1) &= c(\text{Right}) + (0.8V(s_2) + 0.2V(s_1) + 0V(s_3)) = 1 \\
V(s_2) &= c(\text{Right}) + (0.8V(s_3) + 0.2V(s_2) + 0V(s_1)) = 1 \\
\text{Iteration 1: } V(s_1) &= 1 + (0.8 \times 0 + 0.2 \times 0 + 0 \times 0) = 1 \\
V(s_2) &= 1 + (0.8 \times 0 + 0.2 \times 0 + 0 \times 0) = 1 \\
\text{Iteration 2: } V(s_1) &= 1 + (0.8 \times 1 + 0.2 \times 1) = 2 \\
V(s_2) &= 1 + (0.8 \times 0 + 0.2 \times 1) = 1.2 \\
\text{Iteration 3: } V(s_1) &= 1 + (0.8 \times 1.2 + 0.2 \times 2) = 2.36 \\
V(s_2) &= 1 + (0.8 \times 0 + 0.2 \times 1.2) = 1.24 \\
\text{Iteration 4: } V(s_1) &= 1 + (0.8 \times 1.24 + 0.2 \times 2.36) = 2.464 \\
V(s_2) &= 1 + (0.8 \times 0 + 0.2 \times 1.24) = 1.248 \\
\text{Iteration 5: } V(s_1) &= 1 + (0.8 \times 1.248 + 0.2 \times 2.464) = 2.4912 \\
V(s_2) &= 1 + (0.8 \times 0 + 0.2 \times 1.248) = 1.2496 \\
\text{Iteration 6: } V(s_1) &= 1 + (0.8 \times 1.2496 + 0.2 \times 2.4912) = 2.49792 \\
V(s_2) &= 1 + (0.8 \times 0 + 0.2 \times 1.2496) = 1.24992
\end{aligned}
$$

Termination on Value Iteration

- Value Iteration ends when it reaches a *fixed point*, i.e., when values do not change from one iteration to the next
- In practice, it stops when $\max_s \mid V(s)^t - V(s)^{t+1} \mid \leq \epsilon$
- For $\epsilon = 0.1$ the previous example stops after Iteration 5
- For $\epsilon = 0.01$ the previous example does not stop after Iteration 6

Outline

1 Introduction

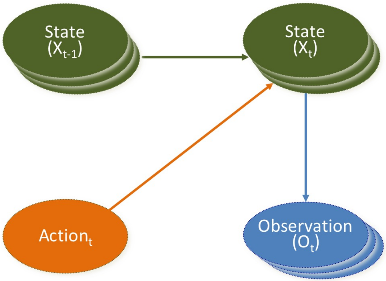
2 Probabilistic reasoning

3 Bayesian networks

4 Markov Models
 - Introduction
 - Markov chains
 - Hidden Markov Models
 - Markov Decision Processes (MDP)
 - Partially Observable MDPs (POMDPs)

5 Fuzzy logic

Partially Observable MDPs (POMDPs)



Markov tasks

- Markov chains
- Markov chains + partial observability = HMM
- Markov chains + actions = MDP
- Markov chains + partial observability + actions = HMM + actions = MDP + partial observability = POMDP

	Observability	
actions	**full**	**partial**
no	Markov chains	HMM
yes	MDP	POMDP

Summary

- To model the notion of probabilities + time we can use variables X_t
- Markov Assumption:
  - states that variables X_t depend solely of variables in t-1
  - allows us to represent the distribution compactly
- We can model non-deterministic actions with probabilities extending the search model
- Solutions are not longer paths but policies
- Bellman equation characterizes the expected value of being in a state
- Expected values allow us to compute optimal policies
- We can obtain these values by an iterative algorithm

Credits

- Material of previous years at UC3M
- Material by Héctor Geffner
- Book and teaching notes of *Artificial Intelligence: A Modern Approach*. Russell&Novig. 2nd edition
- Comparative table:
  `http://www.cassandra.org/pomdp/pomdp-faq.shtml`

Outline

1. Introduction
2. Probabilistic reasoning
3. Bayesian networks
4. Markov Models
5. Fuzzy logic

References