CS 2740 Knowledge representation
Lecture 24

Markov decision processes

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Administrative announcements

Final exam:

- **Monday, December 8, 2008**
- **In-class**
- **Only the material covered after the midterm**

Term project assignment:

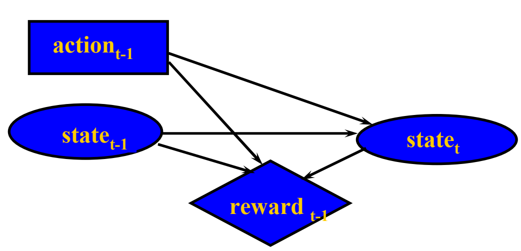
- **Thursday, December 11, 2008 noon**

Decision trees

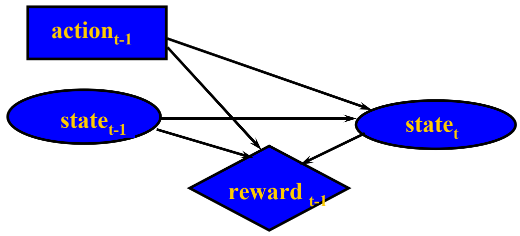
- **Decision tree:**
  – A basic approach to represent decision making problems in the presence of uncertainty
- **Limitations:**
  – What if there are many decision stages to consider?
  – What if there are many different outcomes?
- **Markov decision process (MDP)**
  – A framework for representing complex multi-stage decision problems in the presence of uncertainty
  – More compact representation of the decision problem
  – Efficient solutions

Markov decision process

- **Markov decision process (MDP)**
  – **Models the dynamics of the environment under different actions**
  – Frequently used in AI, OR, control theory
  – **Markov assumption**: next state depends on the previous state and action, and not states (actions) in the past



Markov decision process

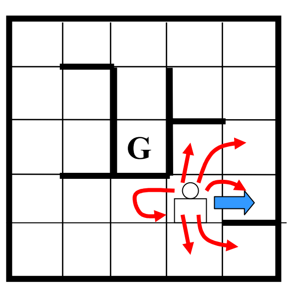


Formal definition: 4-tuple (S, A, T, R)

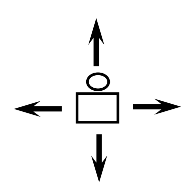
• **A set of states** S (X)	locations of a robot
• **A set of actions** A	move actions
• **Transition model** $S \times A \times S \rightarrow [0,1]$	where can I get with different moves
• **Reward model** $S \times A \times S \rightarrow \Re$	reward/cost for a transition

Markov decision problem

- **Example: agent navigation in the Maze**
  - 4 moves in compass directions
  - Effects of moves are stochastic – we may wind up in other than intended location with non-zero probability
  - **Objective:** reach the goal state in the shortest expected time

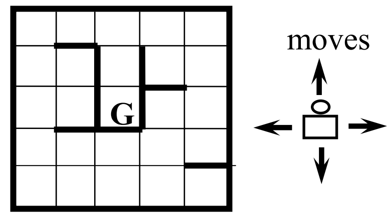


moves



Agent navigation example

- **An MDP model:**
  - **State:** S – position of an agent
  - **Action:** A –a move
  - **Transition model** P
  - **Reward:** R
    - -1 for each move
    - +100 for reaching the goal

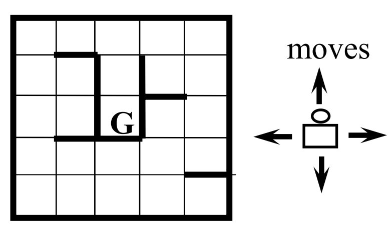


- **Goal: find the policy maximizing future expected rewards**

Policy

- **A policy:**
  maps states to actions

  $$\pi : S \rightarrow A$$



$$\pi : \left| \begin{array}{l} \textbf{Position 1} \longrightarrow \textbf{\textit{right}} \\ \textbf{Position 2} \longrightarrow \textbf{\textit{right}} \\ \ldots \\ \textbf{Position 20} \longrightarrow \textbf{\textit{left}} \end{array} \right|$$

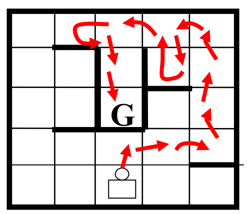
MDP problem

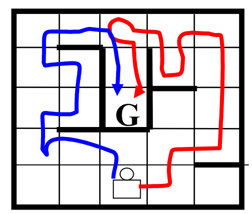
- We want to find the best policy $\pi^*: S \rightarrow A$
- **Value function** (V) for a policy, quantifies the goodness of a policy through, e.g. infinite horizon, discounted model

$$E(\sum_{t=0}^{\infty} \gamma^t r_t)$$

It: 1. combines future rewards over a trajectory

2. combines rewards for multiple trajectories (through expectation-based measures)





Valuation models

- **Objective:**

  Find a policy $\pi^*: S \rightarrow A$

  That maximizes some combination of future reinforcements (rewards) received over time
- **Valuation models (quantify how good the mapping is):**
  - **Finite horizon model**

    $$E(\sum_{t=0}^{T} r_t) \qquad \text{Time horizon: } T > 0$$
  - **Infinite horizon discounted model**

    $$E(\sum_{t=0}^{\infty} \gamma^t r_t) \qquad \text{Discount factor: } 0 < \gamma < 1$$
  - **Average reward**

    $$\lim_{T \rightarrow \infty} \frac{1}{T} E(\sum_{t=0}^{T} r_t)$$

Value of a policy for MDP

- Assume a fixed policy $\pi : S \rightarrow A$
- How to compute the value of a policy under infinite horizon discounted model?

Fixed point equation:

$$V^{\pi}(s) = \underbrace{R(s,\pi(s))}_{\text{expected one step reward for the first action}} + \underbrace{\gamma \sum_{s' \in S} P(s' \mid s, \pi(s)) V^{\pi}(s')}_{\text{expected discounted reward for following the policy for the rest of the steps}}$$

$$\mathbf{v} = \mathbf{r} + \mathbf{U}\mathbf{v} \quad \Rightarrow \quad \mathbf{v} = (\mathbf{I} - \mathbf{U})^{-1}\mathbf{r}$$

– For a finite state space– we get a set of linear equations

Optimal policy

- **The value of the optimal policy**

$$V^{*}(s) = \max_{a \in A} \left[\underbrace{R(s,a)}_{\text{expected one step reward for the first action}} + \underbrace{\gamma \sum_{s' \in S} P(s' \mid s, a) V^{*}(s')}_{\text{expected discounted reward for following the opt. policy for the rest of the steps}} \right]$$

Value function mapping form:

$$V^{*}(s) = (HV^{*})(s)$$

- **The optimal policy:** $\pi^{*} : S \rightarrow A$

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

Computing optimal policy

Three methods:

- **Value iteration**
- **Policy iteration**
- **Linear program solutions**

Value iteration

Method:

- computes the optimal value function first then the policy
- iterative approximation
- converges to the optimal value function

Value iteration (ε **)**

initialize $\mathbf{V}$;; V is vector of values for all states

repeat

set $\mathbf{V}' \leftarrow \mathbf{V}$

set $\mathbf{V} \leftarrow \mathbf{HV}$

until $\left\|\mathbf{V}' - \mathbf{V}\right\|_{\infty} \leq \varepsilon$

output $\pi^*(s) = \arg\max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) V(s') \right]$

Policy iteration

Method:
- Takes a policy and computes its value
- Iteratively improves the policy, till policy cannot be further improved

Policy iteration

initialize a policy μ

repeat

set $\pi \leftarrow \mu$

compute V^{π}

compute $\mu(s) = \arg\max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) V^{\pi}(s') \right]$

until $\pi \equiv \mu$

output π

Linear program solution

Method:
- converts the problem as a linear program problem

- Let v_s denotes V(s)
- Linear program:

$$\max \sum_{s \in S} v_s$$

Subject to:

$$v_s \geq R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) v_{s'} \quad \forall s \in S, a \in A$$