Markov Decision Processes

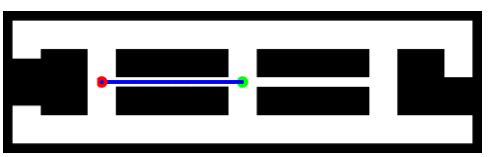
CSE 473

Chapter 17

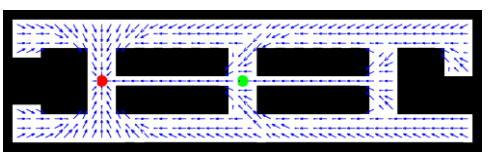
Problem Classes

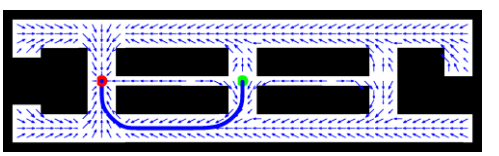
- Deterministic vs. stochastic actions
- Full vs. partial observability

Deterministic, fully observable

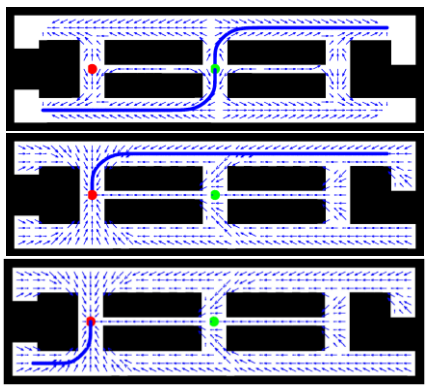


Stochastic, Fully Observable

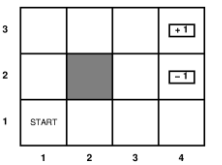




Stochastic, Partially Observable



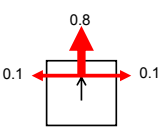
Sequential Decision Problem



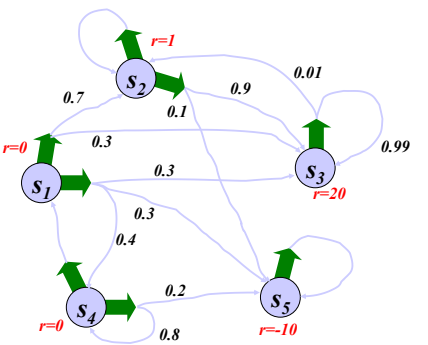
- **Beginning in the start state,** agent must **choose** an **action at each time step**.
- Interaction with environment **terminates** if the **agent reaches one of the goal states** (4, 3) (reward of +1) or (4,1) (reward –1). Each other location has a reward of -.04.
- **In each location** the **available actions** are Up, Down, Left, Right.

Stochastic Actions

- Each action achieves the intended effect with probability 0.8, but the rest of the time, the agent moves at right angles to the intended direction.



Markov Decision Process (MDP)



Markov Decision Process (MDP)

Given a **set of states** in an accessible, stochastic environment, an MDP is defined by

- Initial state S_0
- Transition Model T(s,a,s')
- Reward function R(s)

Transition model: T(s,a,s') is the probability that state s' is reached, if action a is executed in state s.

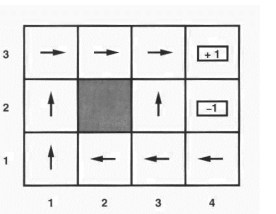
Policy: Complete mapping π that specifies for each state s which action $\pi(s)$ to take.

Wanted: The optimal policy π^* that maximizes the expected utility.

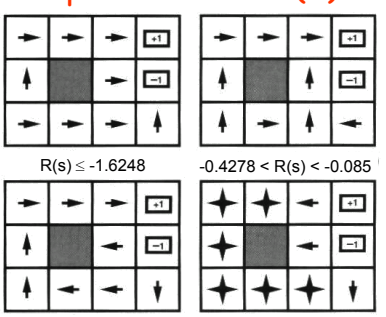
Optimal Policies (1)

- Given the optimal policy, the agent uses its **current percept** that **tells** it its **current state**.
- It then **executes** the **action** $\pi^*(s)$.
- We obtain a simple reflex agent that is computed from the information used for a utility-based agent.

Optimal policy for our MDP:



Optimal Policies (2)



R(s) ≤ -1.6248

-0.4278 < R(s) < -0.085

-0.0221 < R(s) < 0

0 < R(s)

How to compute optimal policies?

Horizon and Rewards

- Finite : Plan t steps into future.
  Reward = $R(s^0)+R(s^1)+R(s^2)+\ldots+R(s^t)$
  Optimal action changes with time!
- Infinite : The agent never dies.
  The reward $R(s^0)+R(s^1)+R(s^2)+\ldots$ could be unbounded.
  - Discounted reward : $R(s^0)+\gamma R(s^1)+ \gamma^2 R(s^2)+\ldots$
  - Average reward : $\lim_{n\to\infty} (1/n)[\Sigma_i R(s^i)]$

Utilities of States

- The **utility of a state** depends on the **utility of the state sequences that follow it**.
- Let $U^{\pi}(s)$ bet the utility of a state under policy π.
- Let s_t be the state of the agent after executing π for t steps. Thus, the utility of s under π is

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} {}^{t}R(s_t) \mid \pi, s_0 = s\right]$$

- The true utility U(s) of a state is $U^{\pi^*}(s)$.
- R(s) is the **short-term reward for being in s** and U(s) is the **long-term total reward from s onwards**.

Example

The utilities of the states with γ=1 and R(s)=-0.04 for nonterminal states:

3	0.812	0.868	0.918	+1
2	0.762		0.660	–1
1	0.705	0.655	0.611	0.388
	1	2	3	4

Choosing Actions using the Maximum Expected Utility Principle

The agent simply chooses the **action that maximizes the expected utility of the subsequent state**:

$$\pi(s) = \arg\max_{a} \sum_{s'} T(s,a,s')U(s')$$

The **utility of a state** is the **immediate reward** for that state **plus** the **expected discounted utility of the next state**, assuming that the agent chooses the optimal action:

$$U(s) = R(s) + \gamma \max_{a} \sum_{s'} T(s,a,s')U(s')$$

Bellman-Equation

- The equation

$$U(s) = R(s) + \gamma \max_{a} \sum_{s'} T(s,a,s')U(s')$$

  is also called the **Bellman-Equation**.
- In our 4x3 world the equation for the state (1,1) is

$$U(1,1) = -0.04 + \gamma \max\{\ 0.8\,U(1,2) + 0.1\,U(2,1) + 0.1\,U(1,1),\ \text{(Up)}$$
$$0.9\,U(1,1) + 0.1\,U(1,2),\ \text{(Left)}$$
$$0.9\,U(1,1) + 0.1\,U(2,1),\ \text{(Down)}$$
$$0.8\,U(2,1) + 0.1\,U(1,2) + 0.1\,U(1,1)\ \}\ \text{(Right)}$$

Given the numbers for the utilities, Up is the optimal action in (1,1).

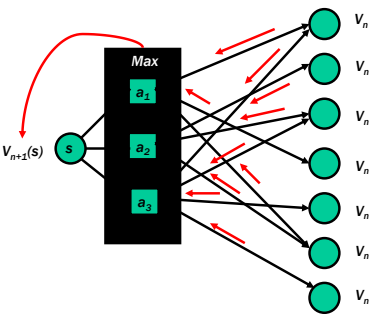
Value Iteration

- The **Bellman equation** is the **basis of value iteration**.
- We can apply an **iterative approach** in which we **replace the equality by an assignment**:

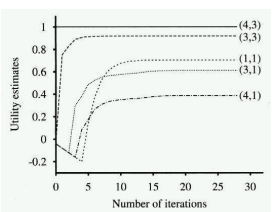
$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s,a,s') U_i(s')$$

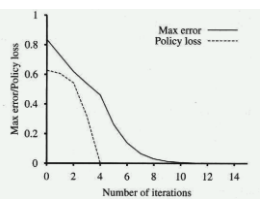
- Update is called Bellman backup
- Stop the iteration appropriately. V_t approaches V* as t increases.

Bellman Backup



Application Example

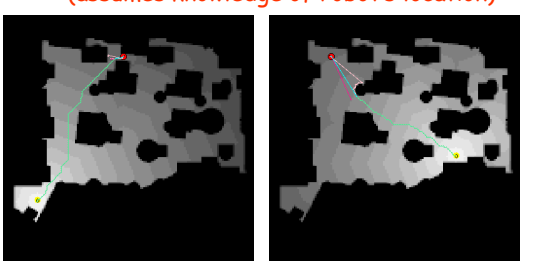




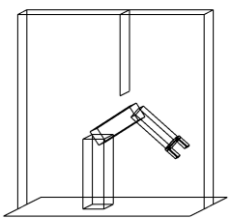
In practice the policy often becomes optimal before the utility has converged.

Value Iteration for Motion Planning

(assumes knowledge of robot's location)



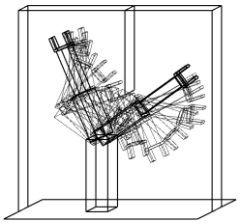
Manipulator Control



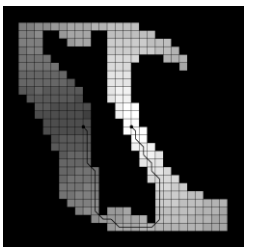
Arm with two joints

Configuration space

Manipulator Control Path

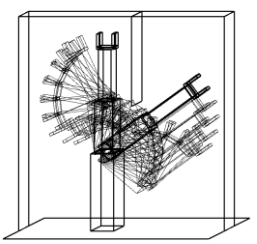


State space



Configuration space

Manipulator Control Path



State space

Configuration space

Complexity of value iteration

- One iteration takes $O(|A||S|^2)$ time.
- Number of iterations required : $\text{poly}(|S|,|A|,1/(1-\gamma))$
- Overall, the algorithm is polynomial in state space, and thus exponential in number of state variables.

Going beyond full observability

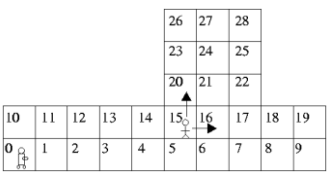
- In execution phase, we are uncertain where we are, but we have some idea of where we can be.
- A belief state = some idea of where we are (represented as a set of/probability distribution over the states).

Partial Observability

- Modelled as POMDPs. (partially observable MDPs). Also called Probabilistic Contingent Planning.
- Belief = probabilistic distribution over states.
- What is the size of belief space?
- Output : Policy (Discretized Belief -> Action)
- Bellman Equation

$$V^*(b)=\max_{a \in A(b)} [c(a)+\Sigma_{o \in O} P(b,a,o)\, V^*(b_a^o)]$$

Example Application



POMDP for People Finding