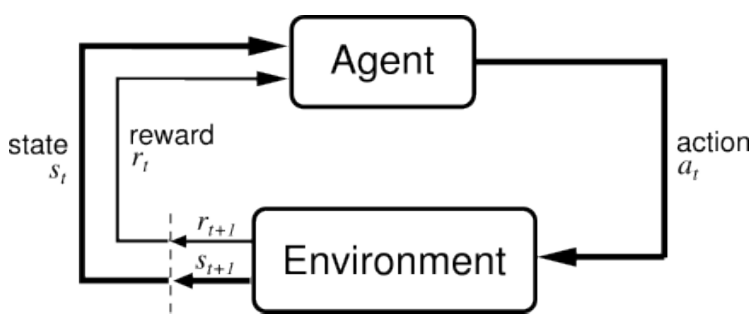
Markov Decision Processes
Value Iteration

Pieter Abbeel
UC Berkeley EECS

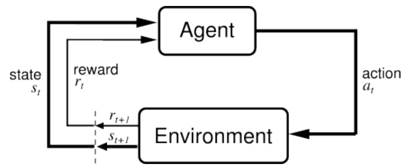
Markov Decision Process



Assumption: agent gets to observe the state

[Drawing from Sutton and Barto, Reinforcement Learning: An Introduction, 1998]

Markov Decision Process (S, A, T, R, H)



Given

- S: set of states
- A: set of actions
- T: S x A x S x {0,1,…,H} → [0,1], $T_t(s,a,s') = P(s_{t+1} = s' \mid s_t = s, a_t = a)$
- R: S x A x S x {0, 1, …, H} → $\Re$ $R_t(s,a,s')$ = reward for ($s_{t+1} = s'$, $s_t = s$, $a_t = a$)
- H: horizon over which the agent will act

Goal:

- Find π : S x {0, 1, …, H} → A that maximizes expected sum of rewards, i.e.,

$$\pi^* = \arg\max_{\pi} \mathrm{E}[\sum_{t=0}^{H} R_t(S_t, A_t, S_{t+1}) | \pi]$$

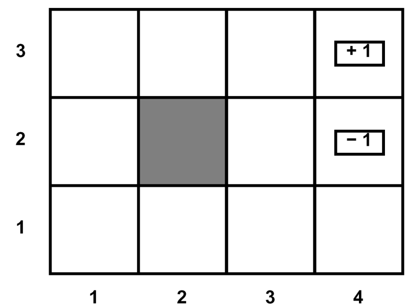
Examples

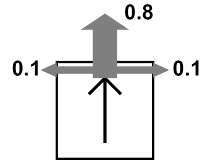
MDP (S, A, T, R, H), goal: $max_{\pi} \mathrm{E}[\sum_{t=0}^{H} R(S_t, A_t, S_{t+1}) | \pi]$

- Cleaning robot
- Walking robot
- Pole balancing
- Games: tetris, backgammon
- Server management
- Shortest path problems
- Model for animals, people

Canonical Example: Grid World

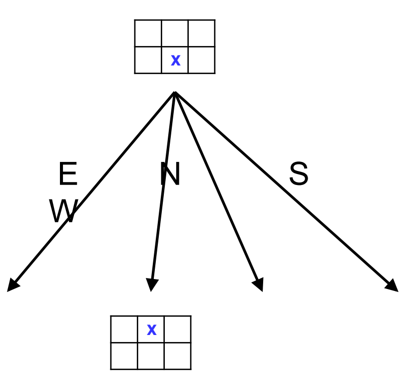
- The agent lives in a grid
- Walls block the agent's path
- The agent's actions do not always go as planned:
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put
- Big rewards come at the end



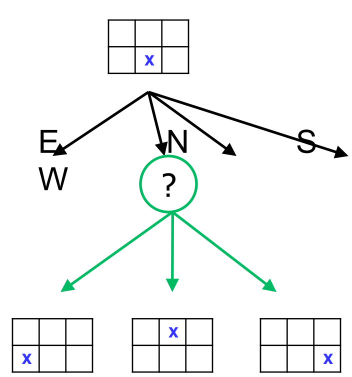


Grid Futures

Deterministic Grid World



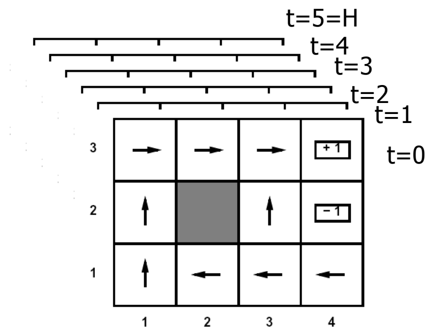
Stochastic Grid World



6

Solving MDPs

- In an MDP, we want an optimal policy π^*: S x 0:H → A
  - A policy π gives an action for each state for each time



  - An optimal policy maximizes expected sum of rewards
- Contrast: In deterministic, want an optimal plan, or sequence of actions, from start to a goal

Value Iteration

- Idea:
  - $V_i^*(s) = \max_{\pi_{H-i:H-1}} \mathrm{E}[\sum_{t=H-i}^{H-1} R_t(S_t, A_t, S_{t+1}) | \pi_{H-i:H}, s_{H-i} = s]$

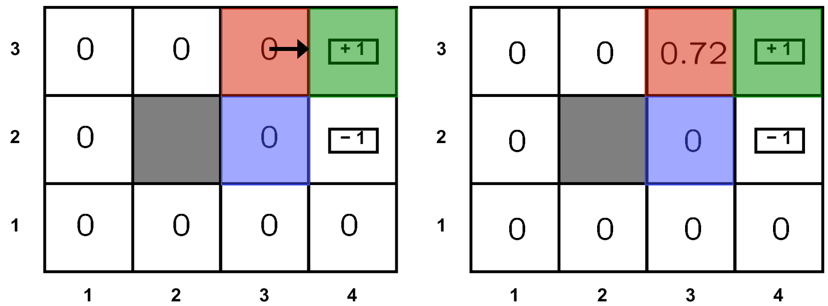
  = the expected sum of rewards accumulated when starting from state s and acting optimally for a horizon of i steps
- Algorithm:
  - Start with $V_0^*(s) = 0$ for all s.
  - For i=1, … , H

    Given V_i^*, calculate for all states s ∈ S:

$$V_{i+1}^*(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + V_i^*(s') \right]$$

  - This is called a value update or Bellman update/back-up

Example

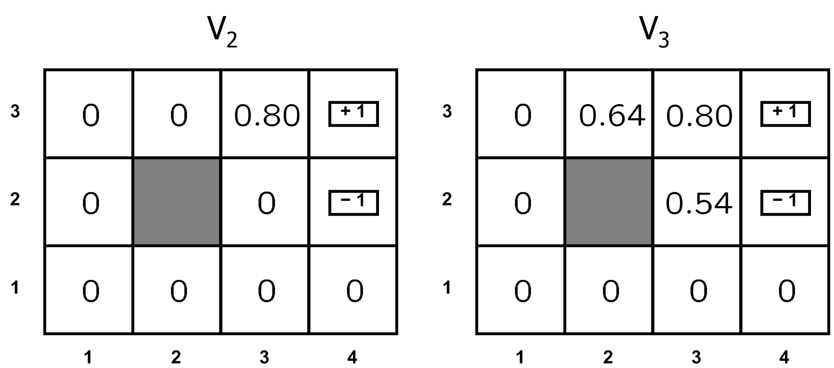


$$V_{i+1}(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + V_i(s')\right]$$

$$V_2(\langle 3,3\rangle) = \sum_{s'} T(\langle 3,3\rangle, \text{right}, s') \left[R(\langle 3,3\rangle) + V_1(s')\right]$$

$$= 0.9\,[0.8 \cdot 1 + 0.1 \cdot 0 + 0.1 \cdot 0]$$

Example: Value Iteration



- Information propagates outward from terminal states and eventually all states have correct value estimates

Practice: Computing Actions

- Which action should we chose from state s:
  - Given optimal values V*?

$$\pi_{H-i}(s) = \arg\max_a \sum_{s'} T_{H-i}(s,a,s')[R_{H-i}(s,a,s') + \gamma V^*_{i-1}(s')]$$

  - = greedy action with respect to V*
  - = action choice with one step lookahead w.r.t. V*

11

Today and forthcoming lectures

- Optimal control: provides general computational approach to tackle control problems.
  - Dynamic programming / Value iteration
    - Discrete state spaces (DONE!)
    - Discretization of continuous state spaces
    - Linear systems
    - LQR
    - Extensions to nonlinear settings:
      - Local linearization
      - Differential dynamic programming
  - Optimal Control through Nonlinear Optimization
    - Open-loop
    - Model Predictive Control
  - Examples: