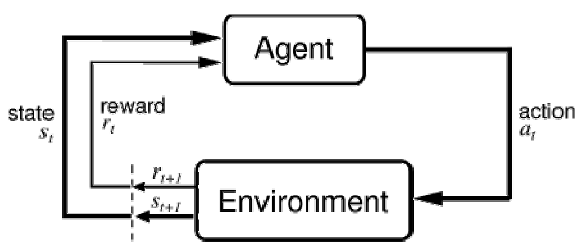
CS 287: Advanced Robotics
Fall 2009

Lecture 11: Reinforcement Learning

Pieter Abbeel
UC Berkeley EECS

Reinforcement Learning



- Model: Markov decision process (S, A, T, R, γ)
  - Goal: Find π that maximizes expected sum of rewards
- T and R might be unknown

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

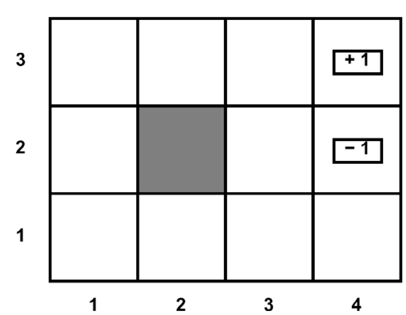
Examples

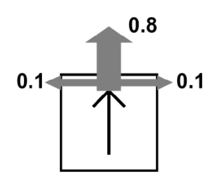
MDP (S, A, T, γ, R), goal: $\max_\pi E [\sum_t \gamma^t R(s_t, a_t) \mid \pi]$

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

Canonical Example: Grid World

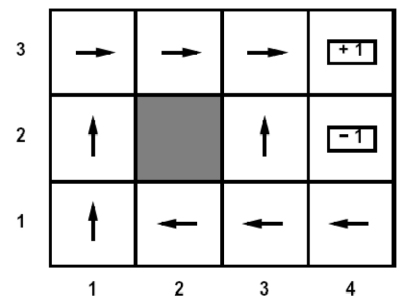
- The agent lives in a grid
- Walls block the agent's path
- The agent's actions do not always go as planned:
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent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



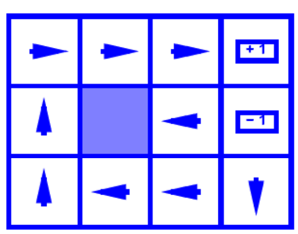


Solving MDPs

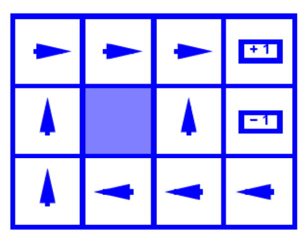
- In deterministic single-agent search problem, want an optimal plan, or sequence of actions, from start to a goal
- In an MDP, we want an optimal policy $\pi^*: S \rightarrow A$
  - A policy π gives an action for each state
  - An optimal policy maximizes expected utility if followed
  - Defines a reflex agent



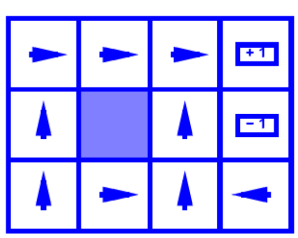
Example Optimal Policies



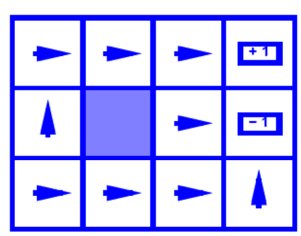
R(s) = -0.02



R(s) = -0.04



R(s) = -0.1



R(s) = -2.0

Outline current and next few lectures

- Recap and extend exact methods
  - Value iteration
  - Policy iteration
  - Generalized policy iteration
  - Linear programming [later]

- Additional challenges we will address by building on top of the above:
  - Unknown transition model and reward function
  - Very large state spaces

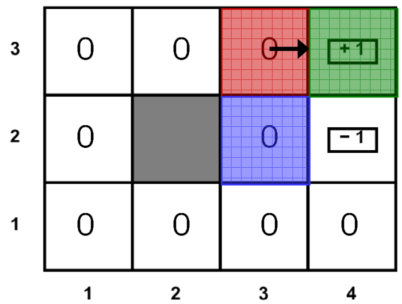
Value Iteration

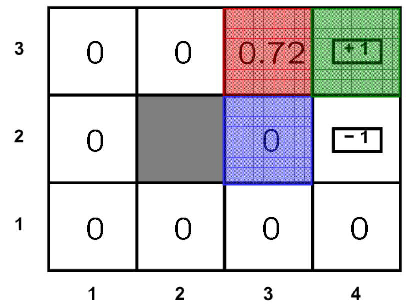
- Algorithm:
  - Start with $V_0(s) = 0$ for all s.
  - Given V_i, calculate the values for all states for depth i+1:

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

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

Example: Bellman Updates





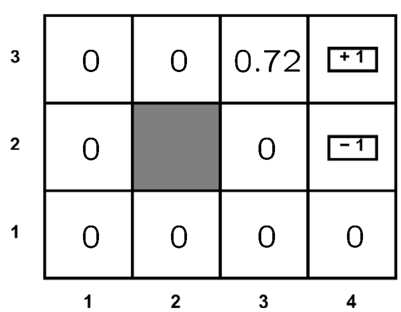
$$V_{i+1}(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

$$V_2(\langle 3,3 \rangle) = \sum_{s'} T(\langle 3,3 \rangle, \text{right}, s') \left[R(\langle 3,3 \rangle) + 0.9\, V_1(s') \right]$$

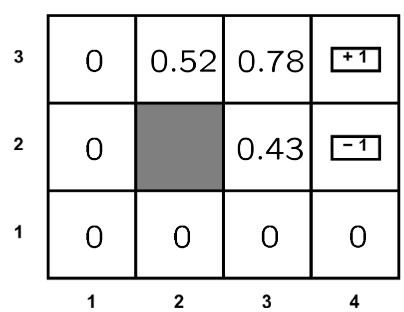
$$= 0.9\, [0.8 \cdot 1 + 0.1 \cdot 0 + 0.1 \cdot 0]$$

Example: Value Iteration

V_2



V_3



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

Convergence

Infinity norm: $\|V\|_\infty = \max_s |V(s)|$

Fact. Value iteration converges to the optimal value function V^* which satisfies the Bellman equation:

$$\forall s \in S: \quad V^*(s) = \max_a \sum_{s'} T(s,a,s')(R(s,a,s') + \gamma V^*(s'))$$

Or in operator notation: $V^* = TV^*$ where T denotes the Bellman operator.

Fact. If an estimate V satisfies $\|V - TV\|_\infty \leq \epsilon$ then we have that

$$\|V - V^*\|_\infty \leq \frac{\epsilon}{1-\gamma}$$

Practice: Computing Actions

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

$$\pi(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

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

12

Policy Iteration

- Alternative approach:
  - Step 1: Policy evaluation: calculate value function for a fixed policy (not optimal!) until convergence
  - Step 2: Policy improvement: update policy using one-step lookahead with resulting converged (but not optimal!) value function
  - Repeat steps until policy converges

- This is policy iteration
  - It's still optimal!
  - Can converge faster under some conditions

13

Policy Iteration

- Policy evaluation: with fixed current policy π, find values with simplified Bellman updates:
  - Iterate until values converge

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

- Policy improvement: with fixed utilities, find the best action according to one-step look-ahead

$$\pi_{k+1}(s) = \arg\max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^{\pi_k}(s')\right]$$

14

Comparison

- Value iteration:
  - Every pass (or “backup”) updates both utilities (explicitly, based on current utilities) and policy (possibly implicitly, based on current policy)
- Policy iteration:
  - Several passes to update utilities with frozen policy
  - Occasional passes to update policies
- Generalized policy iteration:
  - General idea of two interacting processes revolving around an approximate policy and an approximate value
- Asynchronous versions:
  - Any sequences of partial updates to either policy entries or utilities will converge if every state is visited infinitely often