

Markov Decision Processes

CPS 271
Ron Parr

Applications of MDPs

- Economics/Operations Research
 - Fleet maintenance (Howard, Rust)
 - Road maintenance (Golabi et al.)
 - Packet Retransmission (Feinberg et al.)
 - Nuclear plant management (Rothwell & Rust)

Applications of MDPs

- AI/Computer Science
 - Robotic control (Koenig & Simmons, Thrun et al., Kaelbling et al.)
 - Air Campaign Planning (Meuleau et al.)
 - Elevator Control (Barto & Crites)
 - Computation Scheduling (Zilberstein et al.)
 - Control and Automation (Moore et al.)
 - Spoken dialogue management (Singh et al.)
 - Cellular channel allocation (Singh & Bertsekas)

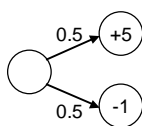
Applications of MDPs

- EE/Control
 - Missile defense (Bertsekas et al.)
 - Inventory management (Van Roy et al.)
 - Football play selection (Patek & Bertsekas)
- Agriculture
 - Herd management (Kristensen, Toft)

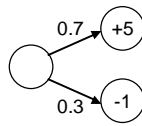


The MDP Framework

- We return to thinking at the level of *states*
- Probabilistic state transitions
- Multiple actions
- Reward function



Action 1



Action 2

How Do MDPs Differ From Search

- In search, we preferred to think about trees because graphs made things complicated
- Would like to use expectimax, but
- When we add probabilities, tree assumption becomes unrealistic
- Need to address loops head on

The MDP Framework

- State space: S
- Action space: A
- Transition function: P
- Reward function: R
- Discount factor: γ
- Policy: $\pi(s) \rightarrow a$

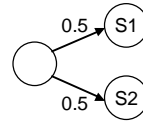
Objective: Maximize expected, discounted return

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

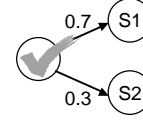
Finding Good Policies

Suppose an expert told you the "value" of each state:

$$V(S1) = 10 \quad V(S2) = 5$$



Action 1



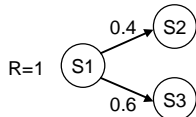
Action 2

Value Determination

Determine the value of each state under policy p

$$V(s) = R(s) + \gamma \sum_{s'} P(s' | s, \pi(s)) V(s')$$

Bellman Equation



$$V(s_1) = 1 + \gamma(0.4V(s_2) + 0.6V(s_3))$$

Matrix Form

$$\mathbf{P} = \begin{pmatrix} P(s_1 | s_1, \pi(s_1)) & P(s_2 | s_1, \pi(s_1)) & P(s_3 | s_1, \pi(s_1)) \\ P(s_1 | s_2, \pi(s_2)) & P(s_2 | s_2, \pi(s_2)) & P(s_3 | s_2, \pi(s_2)) \\ P(s_1 | s_3, \pi(s_3)) & P(s_2 | s_3, \pi(s_3)) & P(s_3 | s_3, \pi(s_3)) \end{pmatrix}$$

$$\mathbf{V} = \gamma \mathbf{P}_{\pi} \mathbf{V} + \mathbf{R}$$

How do we solve this system?

Solving for Values

$$\mathbf{V} = \gamma \mathbf{P}_{\pi} \mathbf{V} + \mathbf{R}$$

For moderate numbers of states we can solve this system exactly:

$$\mathbf{V} = (\mathbf{I} - \gamma \mathbf{P}_{\pi})^{-1} \mathbf{R}$$

Guaranteed invertible because $\gamma \mathbf{P}_{\pi}$ has spectral radius < 1

Solving for Values

$$\mathbf{V} = \gamma \mathbf{P}_{\pi} \mathbf{V} + \mathbf{R}$$

For larger numbers of states we can solve this system indirectly by Gauss iteration:

$$\mathbf{V}_{i+1} = \gamma \mathbf{P}_{\pi} \mathbf{V}_i + \mathbf{R}$$

Guaranteed convergent because $\gamma \mathbf{P}_{\pi}$ has spectral radius < 1

We can also prove that this iteration is a *contraction* in max norm.

Improving Policies

- We can compute the value of a single policy
- How do we get the optimal policy?
- Need to ensure that we take the optimal action in every state:

$$V(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a) V(s')$$

Value Iteration

We can solve the system directly with a max in the equation
Can we solve it iteration?

$$V_{i+1}(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a) V_i(s')$$

- Called *value iteration* or simply *successive approximation*
- We can show that this is also a contraction in max norm
- Guaranteed to converge to optimal policy
- Converges exponentially quickly

Can we do better?

Greedy Policy Construction


Pick action with highest expected future value:

$$\pi(s) = \arg \max_a R(s) + \underbrace{\gamma \sum_{s'} P(s'|s, a) V(s')}_{\text{Expectation over next-state values}}$$

$$\pi = \text{greedy}(V)$$

Bootstrapping: Policy Iteration

Idea: Greedy selection is useful even with suboptimal V

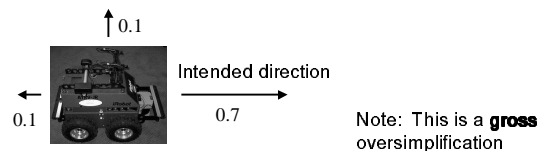
Guess V
 $p = \text{greedy}(V)$
 $V = \text{value of acting on } p$  Repeat until policy doesn't change

Guaranteed to find optimal policy
 Usually takes very small number of iterations
 Computing the value functions is the expensive part

Computational Complexity

- VI and PI are both contraction mappings w/rate γ
 - VI costs less per iteration
 - PI tends to take $O(n)$ iterations in practice
 - Open question: Subexponential bound on PI
- Is there a guaranteed poly time MDP algorithm???

MDP Example



If direction is blocked, stays in some location:



Simple Model

| | | | | |
|---|--|--|-----|----------|
| S | | | +1 | → Absorb |
| | | | -10 | → Absorb |
| | | | | |

Q: What is optimal policy?

Value Iteration (1)

| | | | | |
|---|---|---|-----|----------|
| 0 | 0 | 0 | 1 | → Absorb |
| 0 | | 0 | -10 | → Absorb |
| 0 | 0 | 0 | 0 | |

Start with value 0 for all states
Discount = 0.99

Value Iteration (2)

| | | | | |
|---|---|-------|-------|----------|
| 0 | 0 | 0.69 | 1 | → Absorb |
| 0 | | -0.99 | -10 | → Absorb |
| 0 | 0 | 0 | -0.99 | |

Start with value 0 for all states
Discount = 0.99
Residual = 0.99

Value Iteration (5)

| | | | | |
|------|-------|-------|-------|----------|
| 0.48 | 0.70 | 0.76 | 1 | → Absorb |
| 0.23 | | -0.55 | -10 | → Absorb |
| 0 | -0.20 | -0.23 | -1.40 | |

Start with value 0 for all states
Discount = 0.99
Residual = 0.23

Value Iteration (20)

| | | | | |
|------|------|-------|-------|----------|
| 0.78 | 0.80 | 0.81 | 1 | → Absorb |
| 0.77 | | -0.44 | -10 | → Absorb |
| 0.75 | 0.69 | -0.37 | -0.92 | |

Start with value 0 for all states
Discount = 0.99
Residual = 0.008

Value Iteration (20)

| | | | | |
|------|------|-------|-------|----------|
| 0.78 | 0.80 | 0.81 | 1 | → Absorb |
| 0.77 | | -0.44 | -10 | → Absorb |
| 0.75 | 0.69 | -0.37 | -0.92 | |

Start with value 0 for all states
Discount = 0.99
Residual = 0.008

Final Policy

| | | | |
|------|------|-------|-------|
| 0.78 | 0.80 | 0.81 | 1 |
| 0.77 | | -0.44 | -10 |
| 0.75 | 0.69 | -0.37 | -0.92 |

| | | | |
|---|---|---|-----|
| → | → | → | 1 |
| ↑ | | ↑ | -10 |
| ↑ | ← | ← | ← |

Start with value 0 for all states
Discount = 0.99
Residual = 0.008

Policy Iteration (1)

| | | | |
|---|---|---|-----|
| ↑ | ↑ | ↑ | 1 |
| ↑ | | ↑ | -10 |
| ↑ | ↑ | ↑ | ↑ |

| | | | |
|-------|-------|-------|-------|
| -0.21 | -0.22 | -0.24 | 1 |
| -0.22 | | -1.54 | -10 |
| -0.35 | -1.27 | -2.30 | -8.93 |

Policy Iteration (2)

| | | | |
|---|---|---|-----|
| ↑ | ← | → | 1 |
| ↑ | | ↑ | -10 |
| ↑ | ← | ← | ← |

| | | | |
|------|------|-------|-------|
| 0.37 | 0.41 | 0.76 | 1 |
| 0.36 | | -0.51 | -10 |
| 0.35 | 0.31 | 0.05 | -1.20 |

Policy Iteration (3)

| | | | |
|---|---|---|-----|
| → | → | → | 1 |
| ↑ | | ↑ | -10 |
| ↑ | ← | ← | ← |

| | | | |
|------|------|-------|-------|
| 0.78 | 0.80 | 0.81 | 1 |
| 0.77 | | -0.41 | -10 |
| 0.75 | 0.70 | 0.39 | -0.90 |

Done!

Compare with Value Iteration

| | | | |
|------|------|-------|-------|
| 0.78 | 0.80 | 0.81 | 1 |
| 0.77 | | -0.44 | -10 |
| 0.75 | 0.69 | -0.37 | -0.92 |

Value Iteration
20 Iterations

| | | | |
|------|------|-------|-------|
| 0.78 | 0.80 | 0.81 | 1 |
| 0.77 | | -0.41 | -10 |
| 0.75 | 0.70 | 0.39 | -0.90 |

Policy Iteration
3 Iterations

Change Rewards

| | | | |
|---|---|---|------|
| ↑ | ← | → | 1 |
| → | | ↑ | -500 |
| ↑ | ← | ← | ← |

| | | | |
|-------|-------|-------|-------|
| -2.60 | -3.06 | -6.37 | 1 |
| -2.61 | | -61.9 | -500 |
| -2.85 | -4.82 | -19.1 | -78.2 |

Strange, but optimal.

Linear Programming

$$V(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a) V(s')$$

Issue: Turn the non-linear max into a collection of linear constraints

$$\forall s, a: V(s) \geq R(s) + \underbrace{\gamma \sum_{s'} P(s'|s, a) V(s')}_{\text{Optimal action has tight constraints}}$$

MINIMIZE: $\sum_s V(s)$ Optimal action has tight constraints

Weakly polynomial; slower than PI in practice.

What's The Bad News?

- Works at the level of states = atomic events
- We usually have exponentially many of these
- Hot questions:
 - Combining learning/generalizations with MDPs
 - Solving MDPs when model is not known in advance (Reinforcement Learning)