Markov Decision Processes

CPS 270

Ron Parr

Where We Stand

- Search and planning for deterministic domains
- Efficient manipulation of probabilities
- Decision Theory
  - Act to maximize expected utility
  - How do we do this?
- Markov Decision Processes (MDPs)
  - Algorithmic approach to maximizing utility in situations with loops
  - My favorite topic

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

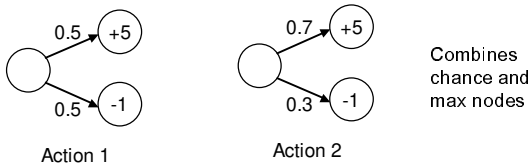
- AI/Computer Science
  - Robotic control (Koenig & Simmons, Thrun et al., Kaelbling et al.)
  - Air Campaign Planning (Meuleau et al.)
  - Elevator Control (Barto & Crites)
  - Computation Scheduling (Zilberstein et al.)
  - Control and Automation (Moore et al.)
  - Spoken dialogue management (Singh 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/cost function defined over states/arcs



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: $\pi(s) \rightarrow a$

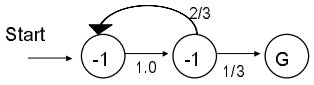
Objective: *Maximize expected, discounted return*

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

The Markov Assumption

- Let S_t be a random variable for the state at time t
- $P(S_t|A_{t-1}S_{t-1},\ldots,A_0S_0) = P(S_t|A_{t-1}S_{t-1})$
- (Use subscripts for time; S0 is different from S_0)
- Markov is special kind of conditional independence
- Future is independent of past given current state

Values of Paths with Loops



- What is the expected cost of this path?
- System of linear equations:

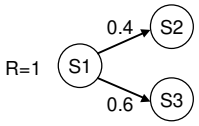
$$V(S1) = -1.0 + V(S2)$$
$$V(S2) = -1.0 + \frac{2}{3}V(S1) + \frac{1}{3}V(G)$$
$$V(G) = 0$$

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

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

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

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 exacty:

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

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

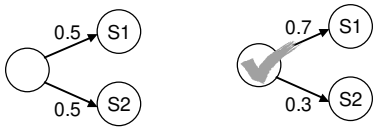
We can also prove that this iteration is a *contraction* in max norm.

Finding Good Policies

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

V(S1) = 10

V(S2) = 5



Action 1 Action 2

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't solve the system directly with a max in the equation
Can we solve it by 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
- Converges exponentially quickly

Can we do better?

Optimality

- VI converges to optimal policy
- Why?
- Optimal policy is stationary
- Why?

Greedy Policy Construction

Pick action with highest expected future value:

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

Expectation over next-state values

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

Bootstrapping: Policy Iteration

Idea: Greedy selection is useful even with suboptimal V

Guess V
π = greedy(V)
V = value of acting on π 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???

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

$$\forall s, a : V(s) \geq \underbrace{R(s) + \gamma \sum_{s'} P(s' | s, a) V(s')}$$

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 research topics:
  - Combining learning/generalizations with MDPs
  - Modeling things at the level of state variables

LAO*

- Combines ideas from A* and MDPs
- Assume admissible heuristic
- Solve interior of graph as MDP
- If all states reachable from initial state under optimal policy are in graph: done
- Otherwise expand fringe node

LAO* Good and Bad

- Good:
  - Can avoid considering entire state space
  - Terminates with optimal solution
- Bad:
  - Hard to give tighter bounds on performance, i.e, no “first path to solution is optimal,” claim
  - Must consider all reachable states from optimal policy – still potentially quite large
- Improvements???