Markov Decision Processes

Rachel Thomas

April 24, 2007
Markov Decision Process

- Model for sequential decision making with uncertainty
- Takes into account both outcome of the current decision and future opportunities
Markov Chain

[Diagram of a Markov Chain with states A, B, C, D, E and transition probabilities labeled on the edges.]
Costs and Rewards

Diagram showing a network of nodes labeled A, B, C, D, and E with assigned costs and rewards:
- Node A: +6
- Node B: +2, +3
- Node C: +3, +5, +2
- Node D: -1, +4
- Node E: +1, -1, -2

Costs:
- A to B: -3
- B to E: -2
- C to D: +5

Rewards:
- A to B: +2
- B to C: +3
- C to D: -1
- D to E: +4
Markov Decision Process $M = (S, A, P_{ss'}, R_{ss'})$

- $S =$ states
- $A =$ actions
- $P_{ss'}^a =$ probability of going from state $s$ to $s'$ when action $a$ is taken
- $R_{ss'}^a =$ reward for going from state $s$ to $s'$ via action $a$

Policy $\pi = \{\pi_1, \pi_2, \ldots\}$

$\pi_i : S \rightarrow A$
Applications

Ex: Inventory management

\[ s = \text{product inventory} \]
\[ a = \text{amt of stock ordered from warehouse} \]
\[ P = \text{random customer demand} \]
\[ \pi = \text{sequence of restocking functions} \]

Other exs: behavioral ecology, gambling, board and computer games, bus engine replacement, communication models
More about MDPs

Markov Decision Process: \( M = \left( S, A, P_{ss'}, R_{ss'}^{a} \right) \)

- Discrete-time dynamic system with transition depending on action \( a \) at state \( s \)
- Reward accumulates additively over time; reward at \( kth \) transition is \( \gamma^k V \), \( 0 < \gamma < 1 \)
- Value function \( V^\pi : S \rightarrow \mathbb{R} \) gives expected long-term discounted sum of rewards when actions chosen according to \( \pi \)
Rewards “diffuse” through state space
Expected immediate reward is

\[ R_{sa} = \sum_{s' \in S} P_{ss'}^a R_{ss'} \]

Discounting factor \( \gamma, 0 < \gamma < 1 \)

\[ V^\pi(s) = R_{s\pi(s)} + \gamma \sum_{s' \in S} P_{ss'}^{\pi(s)} V^{\pi(s)}(s') \]

We’ll assume \( V^\pi < \infty \). Can also study

\[ \lim_{N \to \infty} \frac{1}{N} V_N^\pi \]
The Bellman Operator

Bellman Operator:

$$T^{\pi}(V) = R_{s\pi}(s) + \gamma \sum_{s' \in S} P_{ss'}^{\pi}(s) V(s')$$

Think of $V^{\pi}$ as a vector of dimension $|S|$:

$$V^{\pi} = R^{\pi} + \gamma P^{\pi} V^{\pi}$$

$$V^{\pi} = (I + \gamma P^{\pi} + \gamma^2 (P^{\pi})^2 + \ldots) R^{\pi}$$

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

Our goal is to approximate

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

We define

$$T^*(V) = \max_{\pi} \{R_{s\pi}(s) + \gamma \sum_{s' \in S} P_{ss'}^\pi V(s')\}$$

$V^*$ is the only solution of $V = T^*V$
Optimal value vector satisfies $V^* = T^* V^*$
Start with some $V$ and iterate so that $V_{k+1} = T^* V_k$
And
$$V^* = \lim_{k \to \infty} (T^*)^k V$$
Policy Iteration

Generate a sequence of policies $\pi_1, \pi_2, \ldots$

Given $\pi_k$

1. Policy evaluation step
   Compute $V^{\pi_k} = (I - \gamma P^{\pi_k})^{-1} R^{\pi_k}$

2. (Greedy) policy improvement step
   $\pi_{k+1} = \arg \max_{\pi} \{ R^{s\pi}(s) + \gamma \sum_{s' \in S} P^{\pi}(s') V^{\pi_k}(s')(s') \}$
Since $V \leq V^* = T^* V^*$, $V^*$ is the largest $V$ that satisfies $V \leq T^* V$.

So $V^*(1), \ldots, V^*(n)$ solve

$$\text{maximize } \sum_{s=1}^{n} \lambda_s$$
subject to $$\lambda_s \leq R_{s\pi(s)} + \gamma \sum_{s'=1}^{n} P_{ss'}^{\pi(s)} \lambda_{s'}, \forall \pi$$
Proto-Value Functions

- Traditional methods are using the Euclidean unit orthonormal vectors as a basis for the value space.
- Other methods use a hand-picked basis for the value space. Ex: Chess program- basis functions could include piece mobility, king safety, ...

We want a “better” basis for the space of value functions. *Proto-value functions* (Mahadevan, Maggioni, 2006) form a geometrically customized basis for approximating value functions.
Recall that $V^\pi = (I - \gamma P^\pi)^{-1} R^\pi$.

If $P^\pi$ is diagonalizable,

$$P^\pi = \Phi^\pi \Lambda^\pi (\Phi^\pi)^T,$$

where $\Phi^\pi = (\phi_1^\pi, \ldots, \phi_n^\pi)$

is a complete set of orthonormal eigenvectors

$$P^\pi = \sum_{s=1}^{n} \lambda_s^\pi \phi_s^\pi (\phi_s^\pi)^T$$
$V^{\pi} = \sum_{i=0}^{\infty} (\gamma P^{\pi})^i R^{\pi}$

$V^{\pi} = \sum_{k=1}^{n} \sum_{i=0}^{\infty} \gamma^i (\lambda_k^{\pi})^i \phi_k^{\pi} \alpha_k^{\pi}$

where $R^{\pi} = \Phi^{\pi} \alpha^{\pi}$ and $(P^{\pi})^i \phi_j^{\pi} = (\lambda^{\pi})^i \phi_j^{\pi}$

$V^{\pi} = \sum_{k=1}^{n} \frac{\alpha_k^{\pi}}{1 - \gamma \lambda_k^{\pi}} \phi_k^{\pi}$
Approximation

\[ V^\pi = \sum_{k=1}^{n} \frac{\alpha_k^\pi}{1 - \gamma \lambda_k^\pi} \phi_k^\pi \]

Truncate by choosing \( m < n \) of the eigenvectors. \( \lambda_k^\pi \leq 1 \) so

\[ \frac{1}{1 - \gamma \lambda_k^\pi} \]

is largest when \( \lambda_k^\pi \) is largest.
A few problems

Problems...

• $\pi$ keeps changing.
• $P^{\pi}$ might not be symmetric.
• $P^{\pi}$ might not even be known.

A solution:

• Let $P$ be a symmetric random walk through the state space.
Recall: Policy Iteration

Generate a sequence of policies $\pi_1, \pi_2, \ldots$

1. Policy evaluation step
   Compute $V^{\pi_k} = (I - \gamma P^{\pi_k})^{-1} R^{\pi_k}$

2. (Greedy) policy improvement step
   $\pi_{k+1} = \arg \max_\pi \{ R^{s\pi}(s) + \gamma \sum_{s' \in S} P_{ss'}^{\pi}(s) V^{\pi_k}(s)(s') \}$

$\pi_0(s) = \text{draw uniformly from the possible actions at } s$
(Mahadevan, Maggioni, 2006)
Unified approach to learning representation and behavior:

1. sample collection
2. basis construction
3. policy learning
Iterate
Benefits of Proto-Value Functions

- customized to the geometry of the space
- good when system dynamics and reward function are unknown
Sources

• Bertsekas, Tsitsiklis. *Neuro-Dynamic Programming* (1996)
• Mahadevan, Maggioni. “Proto-Value Functions: A Laplacian Framework for Learning Representation and Control in Markov Decision Processes” (2006)
• Puterman. *Markov Decision Processes* (1994)