Markov Decision Processes

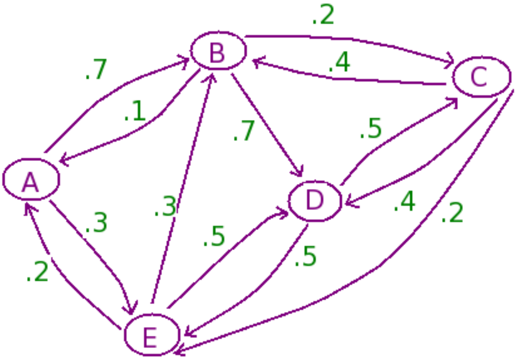
Rachel Thomas

April 24, 2007

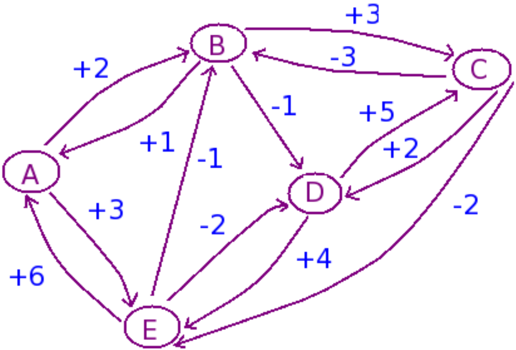
Markov Decision Process

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

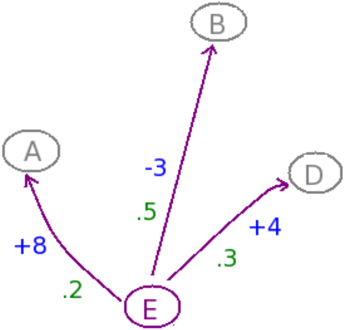
Markov Chain



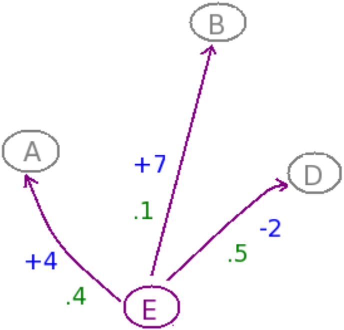
Costs and Rewards



Actions



or



Definition

Markov Decision Process $M = (S, A, P^a_{ss'}, R^a_{ss'})$

$$
\begin{aligned}
S &= \text{states} \\
A &= \text{actions} \\
P^a_{ss'} &= \text{probability of going from state s to s' when action a is taken} \\
R^a_{ss'} &= \text{reward for going from state s to s' via action a}
\end{aligned}
$$

Policy $\pi = \{\pi_1, \pi_2, \dots\}$
$\pi_i : S \to A$

Applications

Ex: Inventory management

$$
\begin{aligned}
s &= \text{product inventory} \\
a &= \text{amt of stock ordered from warehouse} \\
P &= \text{random customer demand} \\
\pi &= \text{sequence of restocking functions}
\end{aligned}
$$

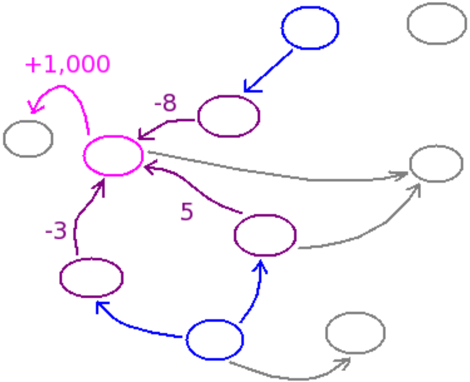
Other exs: behavioral ecology, gambling, board and computer games, bus engine replacement, communication models

More about MDPs

Markov Decision Process: $M = (S, A, P^a_{ss'}, R^a_{ss'})$

- 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 π

Rewards “diffuse” through state space



Expected immediate reward is

$$R_{sa} = \sum_{s' \in S} P^a_{ss'} R^a_{ss'}$$

Discounting factor $\gamma, 0 < \gamma < 1$

$$V^\pi(s) = R_{s\pi(s)} + \gamma \sum_{s' \in S} P^{\pi(s)}_{ss'} V^{\pi(s)}(s')$$

We'll assume $V^\pi < \infty$. Can also study

$$\lim_{N \to \infty} \frac{1}{N} V^\pi_N$$

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^π as a vector of dimension $|S|$:

$$\begin{aligned} V^\pi &= R^\pi + \gamma P^\pi V^\pi \\ V^\pi &= (I + \gamma P^\pi + \gamma^2 (P^\pi)^2 + \dots) R^\pi \\ V^\pi &= (I - \gamma P^\pi)^{-1} R^\pi \end{aligned}$$

Our Goal

Our goal is to approximate

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

We define

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

V^* is the only solution of $V = T^* V$

Value Iteration

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 π_k

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

Linear Programming

Since $V \leq V^* = T^* V^*$, V^* is the largest V that satisfies $V \leq T^* V$.
So $V^*(1), \ldots, V^*(n)$ solve

$$\begin{array}{ll} \text{maximize} & \sum_{s=1}^{n} \lambda_s \\ \text{subject to} & \lambda_s \leq R_{s\pi(s)} + \gamma \sum_{s'=1}^{n} P_{ss'}^{\pi(s)} \lambda_{s'}, \forall \pi \end{array}$$

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 functins. *Proto-value functions* (Mahadevan, Maggioni, 2006) form a geometrically customized basis for approximating value functions.

Eigenfunctions

Recall that $V^\pi = (I - \gamma P^\pi)^{-1} R^\pi$.
If P^π is diagonalizable,

$$
\begin{aligned}
P^\pi &= \Phi^\pi \Lambda^\pi (\Phi^\pi)^T, \\
&\quad \text{where } \Phi^\pi = (\phi_1^\pi, \ldots, \phi_n^\pi) \\
&\quad \text{is a complete set of orthonormal eigenvectors} \\
P^\pi &= \sum_{s=1}^{n} \lambda_s^\pi \phi_s^\pi (\phi_s^\pi)^T
\end{aligned}
$$

V as a combination of eigenvectors

$$
\begin{aligned}
V^\pi &= \sum_{i=0}^{\infty} (\gamma P^\pi)^i R^\pi \\
V^\pi &= \sum_{k=1}^{n} \sum_{i=0}^{\infty} \gamma^i (\lambda_k^\pi)^i \phi_k^\pi \alpha_k^\pi \\
&\quad \text{where } R^\pi = \Phi^\pi \alpha^\pi \text{ and } (P^\pi)^i \phi_j^\pi = (\lambda^\pi)^i \phi_j^\pi \\
V^\pi &= \sum_{k=1}^{n} \frac{\alpha_k^\pi}{1 - \gamma \lambda_k^\pi} \phi_k^\pi
\end{aligned}
$$

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 λ_k^π is largest.

A few problems

Problems...

- π keeps changing.
- P^{π} might not be symmetric.
- P^{π} might not even be known.

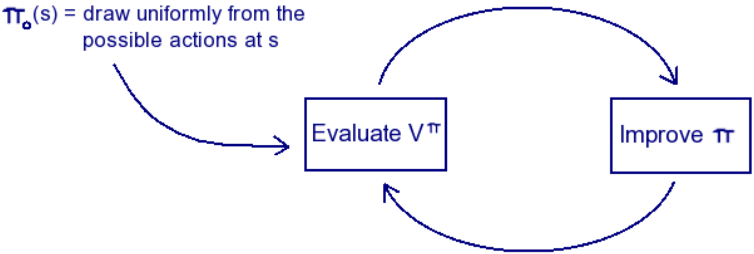
A solution:

- 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
   $\pi_{k+1} = \arg\max_{\pi} \{ R^{s\pi(s)} + \gamma \sum_{s' \in S} P_{ss'}^{\pi(s)} V^{\pi_k(s)}(s') \}$



Representation Policy Iteration

(Mahadevan, Maggioni, 2006)

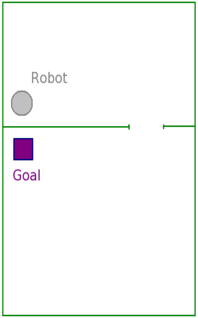
Unified approach to learning represenation and behavior:

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)