Markov Decision Processes

- Markov Decision Processes (MDP)
- MDP solution methods
  - Value Iteration
  - Policy Iteration
  - Monte-Carlo Methods
- Extensions

Markov Decision Process (MDP)

- S = Finite set of states ($|S| = n$)
- A = Finite set of actions ($|A| = m$)
- $\Pr(s_t, a, s_{t+1})$ = Transition function (τ)
  - At each time step the agent observes state $s_t \in S$ and chooses action $a \in A$
  - Represented by set of stochastic matrices (non-deterministic)
    - Also includes events – occurrences not caused by the agent
- $R(s_t, a, s_{t+1})$ = Reward/cost function
- *Markov assumption*
  - Current state s_t encapsulates all previous state and action history $h = (s_0, a_0, \ldots s_{t-1}, a_{t-1})$
  - Optimal action only depends on the current state and action a_t (and in some cases resultant state s_{t+1})
  - Functions $\Pr(s_t, a, s_{t+1})$ and $R(s_t, a, s_{t+1})$ not necessarily known to agent (model free reinforcement learning)

MDP (cont'd)

- Goal
  - Minimize costs
  - Maximize profits
  - Reach goal
- Solution is a *policy* (π^*)
  - Defines the best action a_t to take from every state s_t to maximize future expected utility (reach goals)

Policies

- Nonstationary policy
  - $\pi : S\mathrm{x}T \rightarrow A$
  - $\pi(s,t)$ is action to do at state s with t-stage-to-go
- Stationary Policy
  - $\pi : S \rightarrow A$
  - $\pi(s)$ is action to do at state s (regardless of time) – a universal plan
- Assumed properties
  - Full observability
  - History-independent
  - Deterministic action choice

Value of a Policy

- Value function $V{:}S \rightarrow R$ associates value with each state (sometimes SxT)
- $V_\pi(s)$ denotes value of policy at state s
  - Expected accumulated reward over horizon of interest
  - Note $V_\pi(s)$!= $R(s)$: expected utility
- Common formulations of value:
  - Finite horizon n: total expected reward given π
  - Infinite horizon discounted
  - Infinite horizon – average reward per time step

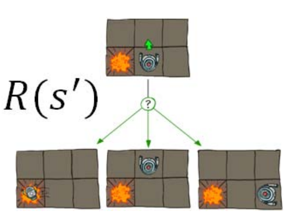
Finite Horizon Value Problems

- Utility (value) depends on stage-to-go
- $V_\pi^k(s)$ is k-stage-to-go value function for π

$$V_\pi^k(s) = E\left[\sum_{t=0}^{k} R^t \middle| \pi, s\right]$$

- R^t is the reward received at stage t
- A single step problem would be:

$$\hat{U}(s,a) = \sum_{s'} \Pr(s,a,s')R(s')$$



Successive Approximation via the Bellman Equation

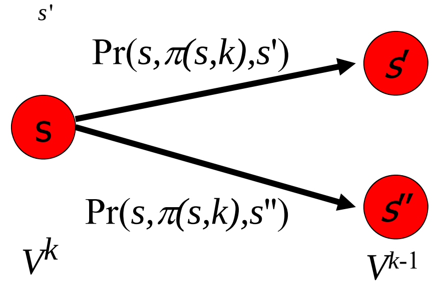
- Successive approximation computes $V_\pi(s)$ by dynamic programming

$$V_\pi^0(s) = R(s)$$

$$V_\pi^k(s) = R(s) + \sum_{s'} \Pr(s, \pi(s,k), s') \cdot V_\pi^{k-1}(s')$$

Policy: pick best action

$$\pi(s,k) = \max_a \Pr(s, a, s')$$



Value Iteration

- Markov property allows exploitation of Dynamic Programming (DP) for optimal policy construction

$$V_\pi^0(s) = R(s)$$

$$V_\pi^k(s) = R(s) + \max_a \sum_{s'} \Pr(s, a, s') \cdot V_\pi^{k-1}(s')$$

$$\pi^*(s,k) = \arg\max_a \sum_{s'} \Pr(s, a, s') \cdot V^{k-1}(s')$$

Actions: (NSEW)

Reward state is accepting (no leaving)

Pr(s,a,s') 0.8 for direction intended, 0.2 for unintended, (0.1,0.1) if 3 reachable states

R=0	R=0	R=100
R=0	R=0	R=0

Value Iteration

- Markov property allows exploitation of DP for optimal policy construction

$$V_\pi^0(s) = R(s)$$
$$V_\pi^k(s) = R(s) + \max_a \sum_{s'} \Pr(s,a,s') \cdot V_\pi^{k-1}(s')$$
$$\pi^*(s,k) = \arg\max_a \sum_{s'} \Pr(s,a,s') \cdot V^{k-1}(s')$$

Actions: (NSEW)

Reward state is accepting (no leaving)

Pr(s,a,s') 0.8 for direction intended, 0.2 for unintended, (0.1,0.1) if 3 reachable states

$V_\pi^0=0$	$V_\pi^0=0$	$V_\pi^0=100$
$V_\pi^0=0$	$V_\pi^0=0$	$V_\pi^0=0$

Value Iteration

- Markov property allows exploitation of DP for optimal policy construction

$$V_\pi^0(s) = R(s)$$
$$V_\pi^k(s) = R(s) + \max_a \sum_{s'} \Pr(s,a,s') \cdot V_\pi^{k-1}(s')$$
$$\pi^*(s,k) = \arg\max_a \sum_{s'} \Pr(s,a,s') \cdot V^{k-1}(s')$$

Actions: (NSEW)

Reward state is accepting (no leaving)

Pr(s,a,s') 0.8 for direction intended, 0.2 for unintended, (0.1,0.1) if 3 reachable states

$V_\pi^1=0$	$V_\pi^1=80$	$V_\pi^1=100$
$V_\pi^1=0$	$V_\pi^1=0$	$V_\pi^1=80$

Value Iteration

- Markov property allows exploitation of DP for optimal policy construction

$$V_\pi^0(s) = R(s)$$

$$V_\pi^k(s) = R(s) + \max_a \sum_{s'} \Pr(s,a,s') \cdot V_\pi^{k-1}(s')$$

$$\pi^*(s,k) = \arg\max_a \sum_{s'} \Pr(s,a,s') \cdot V^{k-1}(s')$$

Actions: (NSEW)

Reward state is accepting (no leaving)

Pr(s,a,s') 0.8 for direction intended, 0.2 for unintended, (0.1,0.1) if 3 reachable states

$V_\pi^2=64$	$V_\pi^2=80$	$V_\pi^2=100$
$V_\pi^2=0$	$V_\pi^2=72$	$V_\pi^2=80$

Value Iteration

- Markov property allows exploitation of DP for optimal policy construction

$$V_\pi^0(s) = R(s)$$

$$V_\pi^k(s) = R(s) + \max_a \sum_{s'} \Pr(s,a,s') \cdot V_\pi^{k-1}(s')$$

$$\pi^*(s,k) = \arg\max_a \sum_{s'} \Pr(s,a,s') \cdot V^{k-1}(s')$$

Actions: (NSEW)

Reward state is accepting (no leaving)

Pr(s,a,s') 0.8 for direction intended, 0.2 for unintended, (0.1,0.1) if 3 reachable states

$V_\pi^3=64$	$V_\pi^3=93.6$	$V_\pi^3=100$
$V_\pi^3=70.4$	$V_\pi^3=72$	$V_\pi^3=94.4$

Value Iteration

- Markov property allows exploitation of DP for optimal policy construction

$$V_\pi^0(s) = R(s)$$

$$V_\pi^k(s) = R(s) + \max_a \sum_{s'} \Pr(s,a,s') \cdot V_\pi^{k-1}(s')$$

$$\pi^*(s,k) = \arg\max_a \sum_{s'} \Pr(s,a,s') \cdot V^{k-1}(s')$$

Actions: (NSEW)

Reward state is accepting (no leaving)

Pr(s,a,s') 0.8 for direction intended, 0.2 for unintended, (0.1,0.1) if 3 reachable states

V_π^4=89	V_π^4=93.6	V_π^4=100
V_π^4=70.4	V_π^4=91.9	V_π^4=94.4

Value Iteration

- Markov property allows exploitation of DP for optimal policy construction

$$V_\pi^0(s) = R(s)$$

$$V_\pi^k(s) = R(s) + \max_a \sum_{s'} \Pr(s,a,s') \cdot V_\pi^{k-1}(s')$$

$$\pi^*(s,k) = \arg\max_a \sum_{s'} \Pr(s,a,s') \cdot V^{k-1}(s')$$

Actions: (NSEW)

Reward state is accepting (no leaving)

Pr(s,a,s') 0.8 for direction intended, 0.2 for unintended, (0.1,0.1) if 3 reachable states

V_π^5=89	V_π^5=98.1	V_π^5=100
V_π^5=91.3	V_π^5=91.9	V_π^5=98.4

Value Iteration

- Because of DP, optimal solution to k-1 stage can be used without modification as part of optimal solution to k-stage problem
- Because of finite horizon, policy is nonstationary
- In practice, Bellman backup computed using:

$$Q^k(s,a) = R(s) + \sum_{s'} \Pr(s,a,s') \cdot V^{k-1}(s'), \quad \forall a$$

$$V^k(s) = \max_a Q^k(s,a)$$

Complexity

- T iterations
- At each iteration $|A|$ computations of nxn matrix time n-vector: $O(|A|n^3)$
- Total $O(T|A|n^3)$
- Can exploit sparsity of matrix: $O(|A|n^2)$

Discounted Infinite Horizon MDPs

- Total reward problematic
  - Many policies have infinite expected reward
  - Zero-cost absorbing state MDPs - OK
- Solution: introduce a discount factor $0 \leq \beta \leq 1$
  - Future rewards discounted by β per time step

$$V_\pi^k(s) = E\left[\sum_{t=0}^{k} \beta^t R^t \middle| \pi, s\right]$$

- Value Iteration function:

$$V_\pi^k(s) = R(s) + \beta \max_a \sum_{s'} \Pr(s, a, s') \cdot V_\pi^{k-1}(s')$$

  - Stop when $\left\| V^k - V^{k-1} \right\| \leq \varepsilon$

Policy Iteration

- Idea: given a baseline policy, an improved policy can be computed using roll-out. The improved policy can be further improved by applying roll-out again. Repeat.
- Since there are a finite number of states and a finite number of actions, this will eventually terminate with a policy that cannot be further improved
- This is in fact an optimal policy.

Policy Iteration

- Given fixed policy, can compute its value exactly:

$$V_\pi(s) = R(s) + \beta \max_a \sum_{s'} \Pr(s, \pi(s), s') \cdot V_\pi(s')$$

- Policy iteration exploits this:

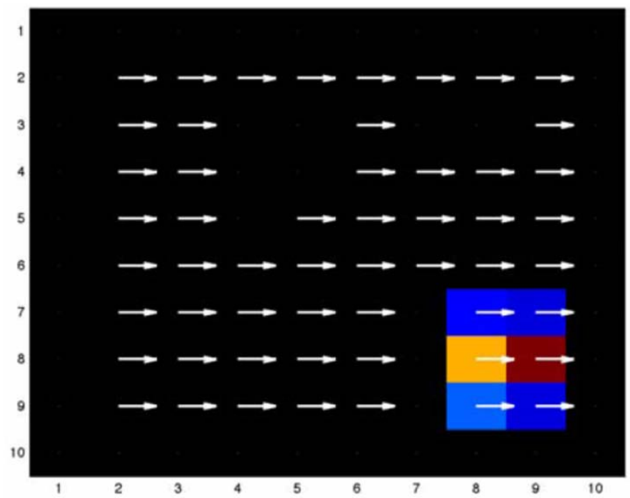
1. Choose a random policy π
2. Loop :
   a. Evaluate V_π
   b. For each s in S, set $\pi'(s) = \arg\max_a \sum_{s'} \Pr(s, a, s') \cdot V_\pi(s)$
   c. Replace π with π'

Until no improving action possible at any state

Policy Iteration

- Pick an arbitrary policy π
- Iterate:
  - Policy Evaluation: Solve for $\forall s \in S$
    - $V(s) = \sum_{s' \in S} Pr(s, \pi(s,a), s')[R(s, \pi(s,a), s') + \gamma V(s')]$
  - Policy Improvement: For each $s \in S$
    - $\pi(s) = \max_a \sum_{s' \in S} Pr(s, \pi(s,a), s')[R(s, \pi(s,a), s') + \gamma V(s')]$
  - Until π is unchanged

Policy Iteration Example

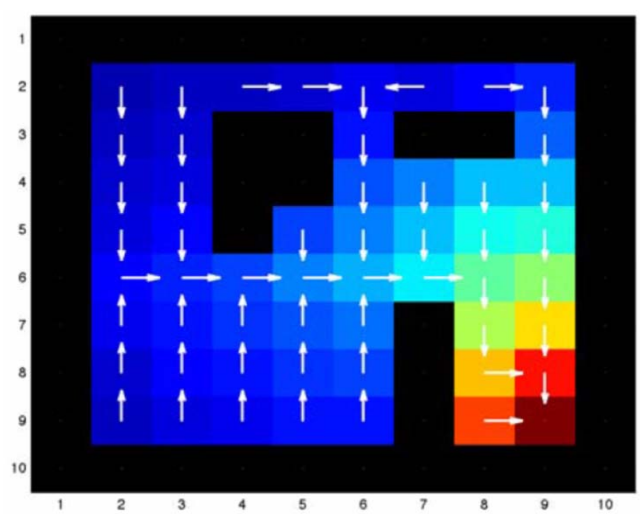


Noise = 0.25

Discount = 0.9

Bottom Left Reward = 1.0

Policy Iteration Example

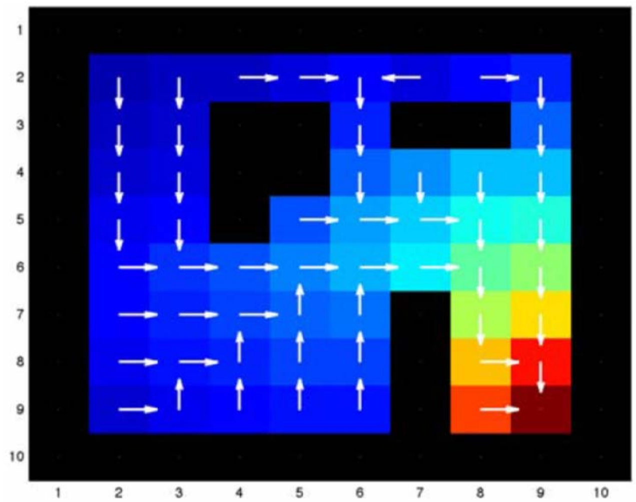


Noise = 0.25

Discount = 0.9

Bottom Left Reward = 1.0

Policy Iteration Example

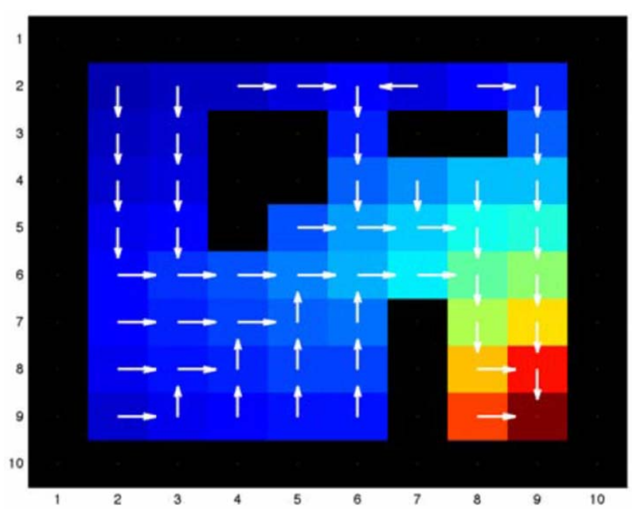


Noise = 0.25

Discount = 0.9

Bottom Left Reward = 1.0

Policy Iteration Example



Noise = 0.25

Discount = 0.9

Bottom Left Reward = 1.0

Policy Iteration Notes

- Convergence assured (Howard)
  - intuitively: no local maxima in value space, and each policy must improve value; since finite number of policies, will converge to optimal policy
- Gives exact value of optimal policy
- Each iteration: $O(|S|^3)$
- Total $O(|A||S|^2+|S|^3)$
- Generally converges much faster than VI
  - each iteration more complex, but fewer iterations
  - Finite-time to convergence

Policy Iteration Example

- After 4 Iterations
- $V(s)=$

0	0	0	0	0	0	0	0	0	0
0	0.45	0.56	0.61	0.84	1.17	0.87	1.11	1.5411	0
0	0.61	0.71	0	0	1.54	0	0	2.16	0
0	0.78	0.93	0	0	2.16	2.59	3.02	3.03	0
0	0.98	1.21	0	2.03	2.74	3.26	3.84	3.91	0
0	1.23	1.58	1.90	2.44	2.95	3.54	4.56	5.03	0
0	1.18	1.50	1.78	2.09	2.28	0	5.38	6.51	0
0	1.02	1.29	1.52	1.76	1.77	0	6.74	8.49	0
0	0.76	1.02	1.20	1.37	1.30	0	8.01	10	0
0	0	0	0	0	0	0	0	0	0

Partially Observable Markov Decision Process (POMDP)

- Assumed that the state was known for each action
  - What about stochastic sending, and unobservable states (fog of war)?

- S = Finite set of states ($|S| = n$)
- A = Finite set of actions ($|A| = m$)
- $\Pr(s_t, a, s_{t+1})$ = Transition function (τ)
  - At each time step the agent observes state $s_t \in S$ and chooses action $a \in A$
  - Represented by set of stochastic matrices (non-deterministic)
    - Also includes events – occurrences not caused by the agent
- $R(s_t, a, s_{t+1})$ = Reward/cost function
- O (or Ω) = Finite set of observations
  - $\Pr(a, s_{t+1}, o)$ = Observation transition function (if Ω this is O)

Decentralized-POMDP (DEC-POMDP)

- Decentralized-POMDP $\{I, S, \{A_i\}, \tau, \{\Omega_i\}, O, R\}$
  - Extends the POMDP to multiple agents
  - All agents have the same reward function
  - Have alternative observations and actions
- I = number of agents
- Joint actions and observations
  - $\vec{A} = \otimes_{i \in I} A_i$ is the set of joint actions, $\vec{a} = \langle a_1, \ldots, a_n \rangle$
  - $\vec{O} = \otimes_{i \in I} O_i$ is the set of joint observations, $\vec{o} = \langle o_1, \ldots, o_n \rangle$

Summary Markov Decision Processes

- Different Learning Problem: Learn policy for action selection that maximizes future reward
- Solution: Learn value function
- Two approaches:
  - Value Iteration
  - Policy Iteration