Markov Decision Process II

Sanjiban Choudhury
Learning

Robot Decision Making

Today!
Recap

Markov Decision Process

A mathematical framework for modeling sequential decision making

< S, A, C, T >

Value of a state-action

\[ Q^\pi(s_t, a_t) = c_t + \gamma c_{t+1} + \gamma^2 c_{t+2} + \cdots \]

Expected discounted sum of cost from starting at a state, executing action and following a policy from then on

\[ Q^\pi(s_t, a_t) = c(s_t, a_t) + \gamma E_{s_{t+1} \sim \mathcal{T}(s_t, a_t)} V^\pi(s_{t+1}) \]

Dynamic Programming all the way!

\[ V^\pi(s_t) = \min_{a_t} [c(s_t, a) + V^\pi(s_{t+1})] \]

\[ \pi^*(s_t) = \arg \min_{a} [c(s_t, a) + V^\pi(s_{t+1})] \]
Does value iteration converge?

What is $V^*(s_1)$? What is $V^*(s_2)$?
What is the effect of discount factor?
Activity!
Think-Pair-Share

Think (30 sec): What are some attributes of a hard MDP?

Pair: Find a partner

Share (45 sec): Partners exchange ideas
Policy Iteration
How frequently does the best action change?

Values

Policy
Policy converges faster than the value

Can we iterate over policies?
Policy Iteration

Init with some policy $\pi$

Repeat forever

Evaluate policy

$$V^{\pi}(s) = c(s, \pi(s)) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^{\pi}(s')$$

Improve policy

$$\pi^+(s) = \arg \min_a c(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^{\pi}(s')$$
Init with some policy $\pi$
Iteration 1

\[ V^\pi(s) = c(s, \pi(s)) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^\pi(s') \]  

\[ \pi^+(s) = \arg \min_a c(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^\pi(s') \]
Policy Iteration

\[ V^\pi(s) = c(s, \pi(s)) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^\pi(s') \] 

\[ \pi^+(s) = \arg \min_a c(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^\pi(s') \]
So many questions ....

Q1. Does policy iteration converge to the optimal policy?

Q2. Does it converge faster than value iteration?

Q3. Can we bootstrap policy evaluation?
Q1. Does policy iteration converge to optimal policy?

Stuck at local minima??
Q1. Does policy iteration converge to optimal policy?

Proof has 2 step argument

Step 1: Policy iteration *monotonically* improves

Step 2: Once it reaches the optimal policy, it does not change
Performance Difference Lemma
Q1. Does policy iteration converge to optimal policy?

2 step argument

- Step 1: Policy iteration \textit{monotonically} improves

  All advantages $\leq 0$ implies monotonic performance improvement

- Step 2: Once it reaches the optimal policy, it does not change

  Advantage $\geq 0$ for the optimal policy
PDL answers all ...

In Imitation Learning

In Model Free Reinforcement Learning

In Model Based Reinforcement Learning

How it partners with online learning
So many questions ....

✅ Q1. Does policy iteration converge to the optimal policy?

Q2. Does it converge faster than value iteration?

Q3. Can we bootstrap policy evaluation?
So many questions ....

Q1. Does policy iteration converge to the optimal policy?

Q2. Does it converge faster than value iteration?
   Empirically yes, but no rigorous theory about when ..

Q3. Can we bootstrap policy evaluation?
So many questions ....

Q1. Does policy iteration converge to the optimal policy? ✓

Q2. Does it converge faster than value iteration?
\_(ツ)_/\  Empirically yes, but no rigorous theory about when..

Q3. Can we bootstrap policy evaluation?
✓ Yes => Modified policy iteration
Messing with MDPs
What happens as you increase the slipperiness of bridge?
What happens if you change the cost function?

\[ C(s) = \| s - s_{goal} \| \]
How do you convert a HARD MDP into an EASY one?
Solving continuous MDPs
Just discretize and run your favorite heuristic search (A*)

2D car, 3D UAV
Run some sequential convex trajectory optim.

CHOMP, TrajOPT
Call a (slow) probabilistic planner that “almost surely” gives you the optimum (Don’t hold your breath…)

Deterministic

Stochastic

Local Opt.

Global Opt.

Low Dim

High Dim

A*

A*

RRT*, BIT*

CHOMP, TrajOPT
Dynamic programming, heuristic search

LAO*, RTDP

A*, A*

RRT*, BIT*

CHOMP, TrajOPT
GIVE UP!

LAO*, RTDP

iLQR

RRT*, BIT*

CHOMP, TrajOPT