Markov Decision Process

Sanjiban Choudhury
Announcements

1. Thanks for finishing Assignment 0!

2. Assignment 1 released!

3. Slides, Python notebook released
Learning

Robot Decision Making

Today!
Question from last class:

“Will we only look at discrete actions?”
Calculus to the rescue

*Develop ideas in discrete space, extend to continuous space*

- Generalized Weighted Majority → Normalized Exponentiated Gradient Descent
- Discrete Value Iteration → Algebraic Ricatti Equations
Learning

Robot Decision Making

Today!
Decisions, decisions!

Tetris

Self-driving

Robot Baristas
What makes decision making hard?

Single shot decision making
What makes decision making hard?

Single shot decision making
What makes decision making hard?

Sequential decision making
What makes decision making hard?

How do we *tractably* reason over a sequence of decisions?
Markov to the rescue!

State: statistic of history sufficient to predict the future

Courtesy: Byron Boots
Markov Decision Process

A mathematical framework for modeling sequential decision making

\[ \langle S, A, C, T \rangle \]
State

Sufficient statistic of the system to predict future disregarding the past
Activity!
Think-Pair-Share

Think (30 sec): Example of MDPs with shallow state?
(Current observation good enough)
Example of MDPs with deep state?

Pair: Find a partner

Share (45 sec): Partners exchange ideas
Action

Doing something: Control action / decisions

\[ \langle S, A, C, \mathcal{T} \rangle \]

\[ a \in A \]
The instantaneous cost of taking an action in a state $c(s, a)$.
Examples of non-Markovian cost?
Transition

The next state given state and action

\[ s' = \mathcal{T}(s, a) \quad \text{Deterministic} \]

\[ s' \sim \mathcal{T}(s, a) \quad \text{Stochastic} \]
Examples of non-Markovian dynamics?

Wind correlates disturbance across time
Markov Decision Process → Problem

Includes things to define an optimization problem

Horizon \( T \in \mathbb{N} \)

Discount \( 0 \leq \gamma \leq 1 \)

Return: \( c_0 + \gamma c_1 + \cdots + \gamma^{T-1} c_{T-1} \)

(Costs are more valuable if they happen soon)
Markov Decision Process → Problem

Policy

\[ \pi \in \Pi \]

\[ \pi : s_t \rightarrow a_t \quad \text{(Deterministic)} \]

\[ \pi : s_t \rightarrow P(a_t) \quad \text{(Stochastic)} \]

A function that maps state (and time) to action

Objective Function

\[ \min_{\pi} \mathbb{E} \left[ \sum_{t=0}^{T-1} \gamma^t c(s_t, a_t) \right] \]

Find policy that minimizes sum of discounted future costs
Value of a state

\[ V_\pi(s_t) = c_t + \gamma c_{t+1} + \gamma^2 c_{t+2} + \ldots \]

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

\[ \pi^* = \arg \min_\pi \mathbb{E}_{s_0} V_\pi(s_0) \]
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

\[ Q^\pi(s_t, a_t) = c(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathcal{T}(s_t, a_t)} V^\pi(s_{t+1}) \]
Values matter
Let’s build some intuition!
Case studies
Example 1: Tetris!

Consider the simplified game of Tetris, where randomly falling pieces must be placed on the game board. Each horizontal line completed is cleared from the board and scores points for the player. The game terminates when the board fills up. The game of Tetris can be modeled as a Markov Decision Process.

Figure 1. Example states and transitions for a Tetris scenario with figure from [3].

- **States:** Board configuration (each of $k$ cells can be filled/not filled), current piece (there are 7 pieces total). In this implementation, there are therefore approximately $2^k \times 7$ states. Note: not all configurations are valid, for example, there cannot be a piece floating in the air. This resulting in a smaller number of total valid states.

- **Actions:** A policy can select any of the columns and from up to 4 possible orientations for a total of about 40 actions (some orientation and column combinations are not valid for every piece).

- **Transition Matrix:** A deterministic update of the board plus the selection of a random piece for the next time-step.

- **Cost Function:** There are several options to choose from, including: reward = +1 for each line removed, 0 otherwise; # of free rows at the top; +1 for not losing that round; etc.

Deterministic and Non-Deterministic MDP Algorithms

For Deterministic MDPs the transition model is deterministic or, equivalently, we know with certainty what the next state $x_0$ will be given the current state $x$ and the action $a$. Solving deterministic $< S, A, C, \mathcal{T} >$?
Example 2: Self-driving

\[ \langle S, A, C, \mathcal{F} \rangle \]
Example 3: Coffee making robot

$< S, A, C, T >$

?
Solving MDPs

Image courtesy Dan Klein
Setup

$< S, A, C, T >$

- Two absorbing states: Goal and Swamp
- Cost of each state is 1 till you reach the goal
- Let’s set $T = 30$
What is the optimal value at $T-1$?

\[ V^*(s_{T-1}) = \min_a c(s_{T-1}, a) \]

\[ \pi^*(s_{T-1}) = \arg \min_a c(s_{T-1}, a) \]
What is the optimal value at T-2?

\[ V^*(s_{T-2}) = \min_a [c(s_{T-2}, a) + V^*(s_{T-1})] \]

\[ \pi^*(s_{T-2}) = \arg \min_a [c(s_{T-2}, a) + V^*(s_{T-1})] \]
Dynamic Programming all the way!

\[ V^*(s_t) = \min_a [c(s_t, a) + V^*(s_{t+1})] \]

\[ \pi^*(s_t) = \arg \min_a [c(s_t, a) + V^*(s_{t+1})] \]
Value Iteration

Algorithm 4: Dynamic Programming Value Iteration for computing the optimal value function.

```
Algorithm OptimalValue(x, T)
for t = T - 1, ..., 0 do
    for x ∈ X do
        if t = T - 1 then
            V(x, t) = min_a c(x, a)
        else
            V(x, t) = min_a c(x, a) + \sum_{x' \in X} p(x'|x, a)V(x, t + 1)
        end
    end
end
```

This approach now has complexity $O(|X|^2 |A| T)$. However, since we often don’t have to sum over all $x \in X$ as the probability of transitioning to those states may be 0, this typically reduces to $O(k |X| |A| T)$, where $k$ is the average number of neighbouring states.

In a deterministic problem, of course $k = 1$. If our environment is continuous, the sums above become integrals as we are integrating over the state space.

Infinite Horizon Problems

Recall that when we have a finite horizon, both the optimal value function and the optimal policy are functions of time. However, as $T$ approaches infinity, we expect that the optimal value function and the optimal policy no longer have such dependence on time. Consider, for example, the maze problem above: we would expect the value function to stabilize as the horizon $T$ gets large. Similarly, it would seem surprising to alter our policy at different time steps.

What is the complexity? Deterministic: $S \times A \times T$; Stochastic: $S^2 \times A \times T$; Efficient: $k \times S \times A \times T$.
Why is the optimal policy a function of time?

Pulling the goalie when you are losing and have seconds left ..
To infinity!
Infinite horizon cases

\[ V^*(s_t) = \min_{a_t} [c(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathcal{F}(s_t, a_t)} V^*(s_{t+1})] \]

Fixed point as \( t \to \infty \)

\[ V^*(s) = \min_{a} [c(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{F}(s, a)} V^*(s)] \]
Bellman Equation

\[ V^*(s) = \min_a \left[ c(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{S}(s,a)} V^*(s') \right] \]

Does this converge?

How fast does it converge?
Does value iteration converge?

What is $V^*(s_1)$? What is $V^*(s_2)$?
Markov Decision Process
A mathematical framework for modeling sequential decision making

\(<S, A, C, \mathcal{T}>\)

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

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

Dynamic Programming all the way!

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

\[\pi^*(s_t) = \arg \min_a [c(s_t, a) + V^\pi(s_{t+1})]\]