MDPs and RL

A Very Short (Defense-Oriented) Introduction

Jefferson Huang

Assistant Professor
Operations Research Department
Naval Postgraduate School

NPS Junior Faculty Research Seminar

May 25, 2022

About Me: A Very Short CV

Recent Academic History:

- Applied Math PhD from Stony Brook University (2016)
- Postdoc in Cornell Operations Research (OR) Department (2016-2018)
- Assistant Prof. in NPS OR Department (2018-Present)

Main Research Interests:

- Markov Decision Processes (**MDPs**)
- Dynamic Resource Allocation Problems
- Defense Logistics

Some Recent Thesis Topics:

- Optimizing Supply Blocks for Expeditionary Units (Capt N.C. Anthony, USMC, June 2021)
- Maximally Informative Underwater Sensor Placement (LT E.V. Vargas, USN, September 2022; co-advising with Robert Bassett)
- Monte-Carlo Methods for Naval Aviation Readiness-Based Sparing Optimization (LCDR A.A. Alleman, USN, September 2022; co-advising with Rudy Yoshida)
- Approximate Dynamic Programming Methods for the Dynamic Airlift Routing Problem (LCDR A.J. Cooper, USN, March 2023)

For more, see https://faculty.nps.edu/jefferson.huang/

MDPs Model (Stochastic) Sequential Decision Problems

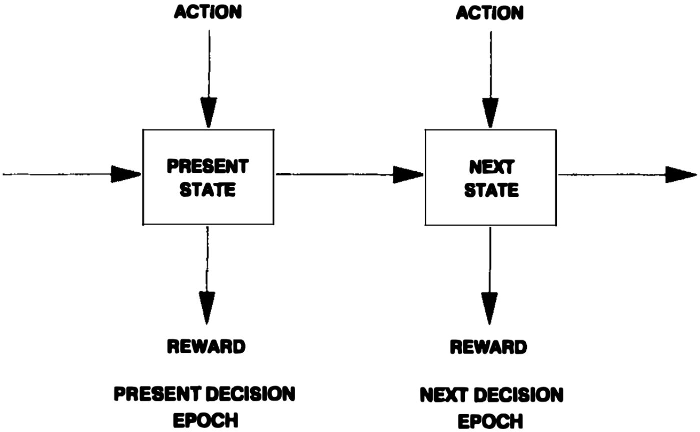


Figure 1.1.1 Symbolic representation of a sequential decision problem.

Source: M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 2005.

Potential defense-related applications abound. . .

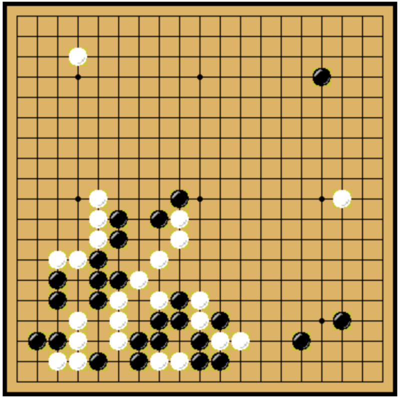
Recent Survey: M. Rempel & J. Cai, A review of approximate dynamic programming applications within military operations research, *Operations Research Perspectives* 8, 2021.

...but computing good solutions at scale is notoriously difficult.

It suffers from what Bellman called "the **curse of dimensionality***," meaning that its computational requirements grow exponentially with the number of state variables . . .*

. . . but it is still far more efficient and more widely applicable than any other general method.

Sutton & Barto, *Reinforcement Learning: An Introduction*, 2018 (p. 14)



http://norvig.com/atoms.html

The MDP Model

A Markov decision process (MDP) is defined by

- a **state set** $\mathbb{X}$,
- sets of feasible **actions** $A(x)$ for each state $x \in \mathbb{X}$,
- **one-step rewards** $r(x, a)$ for each state $x \in \mathbb{X}$ and action $a \in A(x)$, and
- **transition probabilities** $p(y|x, a)$ for $x, y \in \mathbb{X}$ and $a \in A(x)$.

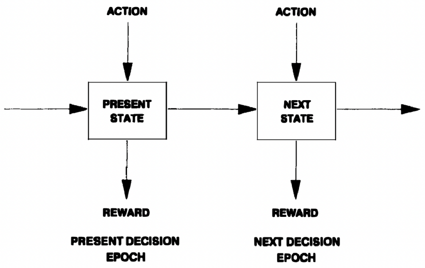


Figure 1.1.1 Symbolic representation of a sequential decision problem.

Source: M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 2005.

Objective: Find an *optimal policy* ϕ that, for each state $x \in \mathbb{X}$, prescribes an action $\phi(x) \in A(x)$ to take.

Finding Optimal Policies

- It suffices to solve some associated **optimality equations** (aka. "Bellman equations")
- Solving the optimality equations provides the **value function**, which indicates which states are more valuable to be in than others.
- The value function can then be used to derive an **optimal policy**.

Example: When the objective is to maximize the expected total reward that is earned, the optimality equations can be viewed as a functional equation

$$v = T(v), \qquad v : \mathbb{X} \to \mathbb{R}$$

where T is a non-linear "optimality operator". Given a solution v to the optimality equation, an optimal policy is

$$\phi^*(x) = \arg\max_{a \in A(x)} \left[r(x,a) + \sum_{y \in \mathbb{X}} v(y) p(y|x,a) \right]$$

Algorithms for Computing Optimal Policies

There are two main algorithmic paradigms:

Value Iteration: Iteratively approximate the value function $v : \mathbb{X} \to \mathbb{R}$.

Example: Start with an initial guess v_0, and iterate the optimality operator T:

$$v_k = T(v_{k-1}), \qquad k = 1, 2, \ldots$$

Policy Iteration: Iteratively approximate the optimal policy ϕ^*.

Example: Start with an initial policy ϕ_0, and iteratively improve it by identifying actions to switch to.

Note: There is a close connection between policy iteration and applying the simplex method to an associated linear program. Policy iteration can also be viewed as applying Newton's method to finding a root of $T(v) - v$. (See e.g., Puterman (2005) for details.)

Dealing with Computational Intractability

There are two main algorithmic paradigms:

Value Function Approximation: Search within a structured classes of value functions.

Examples:

- Piecewise-constant functions (e.g., via state aggregation)
- Parameterized functions (e.g., linear in hand-selected features, neural networks, ...)

Policy Function Approximation: Search within structured classes of policies.

Examples:

- Piecewise-constant policies (e.g., via state aggregation)
- Parameterized policies (e.g., linear in hand-selected features, neural networks, ...)

Reinforcement Learning: Dealing with Unknown/Complex System Dynamics

Idea: Use experience from interacting with an environment (or a simulation model of it) to learn good value or policy function approximations.

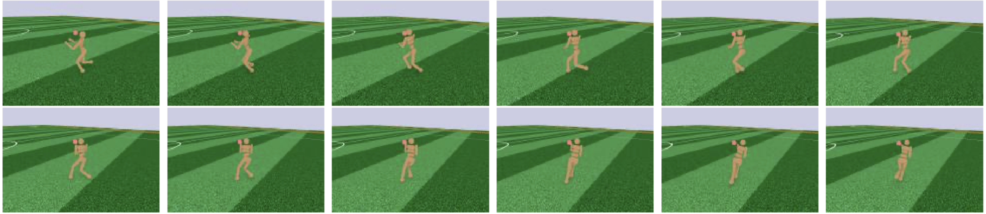


Figure 5: Still frames of the policy learned from RoboschoolHumanoidFlagrun. In the first six frames, the robot runs towards a target. Then the position is randomly changed, and the robot turns and runs toward the new target.

Source: Schulman et al., Proximal policy optimization algorithms, `arXiv:1707.06347v2`, 2017

Reinforcement Learning: Dealing with Unknown/Complex System Dynamics

Idea: Use experience from interacting with an environment (or a simulation model of it) to learn good value or policy function approximations.

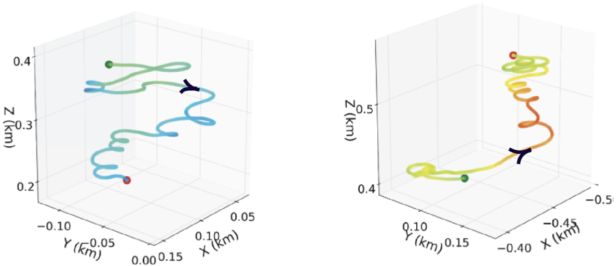


Figure 16.10: Sample thermal soaring trajectories, with arrows showing the direction of flight from the same starting point (note that the altitude scales are shifted). Left: before learning: the agent selects actions randomly and the glider descends. Right: after learning: the glider gains altitude by following a spiral trajectory. Adapted with permission from PNAS vol. 113(22), p. E4879, 2016, Reddy, Celani, Sejnowski, and Vergassola, Learning to Soar in Turbulent Environments.

Source:, Sutton & Barto, 2018 (p. 456)

BLAF (Bottom Line After the Fact)

For MDPs, **modeling is easy** but **computation is hard**.