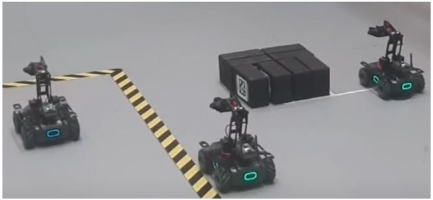
Value Iteration for Learning Concurrently Executable Robotic Control Tasks

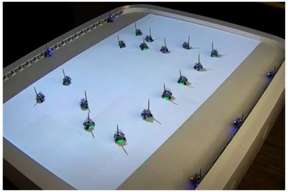
Sheikh A. Tahmid and Gennaro Notomista

24th International Conference on Autonomous Agents and Multiagent Systems

Department of Electrical and Computer Engineering, University of Waterloo, Canada

Motivation





Modern robots are required to do complex tasks and possibly multiple at the same time.

- Let's use RL to learn several control tasks for a robotic system to execute.
  - RL lets us generalize to possibly complex control tasks.
- Let's combine and execute each of these tasks together.
  - Preferably in a way that lets us swap out tasks and/or reorder priorities.
- How do we know that tasks will not interfere with each other?

Assumptions

- Assume that our robotic system is control-affine:

$$\dot{x} = f(x) + g(x)u, \;\; x \in \mathbb{R}^n, \;\; u \in \mathbb{R}^p$$

- Assume that each RL task we learn is encoded with a "cost-to-go"/value function of the form:

$$J_i(x) \approx \min_{u(\cdot)} \int_t^\infty e^{-\beta\tau} \left(q_i(x(\tau)) + \|u(\tau)\|^2 \right) d\tau \qquad\qquad q_i \text{ is P.S.D}$$

Key Related Works

Related Work - Combining Learned Tasks Using a Min-Norm Controller

- Treat learned value functions as Control Lyapunov Functions
- Make progress on each task using constrained optimization problem

$$
\begin{aligned}
\min_{u \in \mathcal{U}, \delta \in \mathbb{R}^N} \quad & \|u\|^2 + \kappa \|\delta\|^2 \\
\text{s.t.} \quad & L_f J_1(x) + L_g J_1(x) u \leq -\sigma_1(x) + \delta_1 \\
& \vdots \\
& L_f J_N(x) + L_g J_N(x) u \leq -\sigma_N(x) + \delta_N \\
& K\delta \geq 0
\end{aligned}
$$

Note that: $L_f J_i(x) = \frac{\partial J_i}{\partial x} f(x)$, $L_g J_i(x) = \frac{\partial J_i}{\partial x} g(x)$.

---

[1] G. Notomista, "A Constrained-Optimization Approach to the Execution of Prioritized Stacks of Learned Multi-robot Tasks," in Distributed Autonomous Robotic Systems, 2024, pp. 479–493

Related Work - Value Iteration for Continuous Action Spaces

Assume continuous, control-affine dynamics and cost function as mentioned previously.

$$\dot{x} = f(x) + g(x)u, \; x \in \mathbb{R}^n, \; u \in \mathbb{R}^p$$

$$J_i(x) \approx \min_{u(\cdot)} \int_t^\infty q_i(x(\tau))) + \|u(\tau)\|^2 d\tau$$

Use expression from solved HJB equation as "optimal input" at each iteration.

$$u^\star = -\frac{1}{2}(L_g J_i(x))^\top = -\frac{1}{2} g(x)^\top \left(\frac{\partial J_i}{\partial x}\right)^\top$$

[2] M. Lutter, S. Mannor, J. Peters, D. Fox, and A. Garg, "Value Iteration in Continuous Actions, States and Time," in Proceedings of the 38th International Conference on Machine Learning, Jul. 2021, vol. 139, pp. 7224–7234.

Related Work - Independence and Orthogonality for Robotic Control Tasks

Come back to this. Finish the other slides first.

How do we know that tasks are compatible with each other?

$$\begin{aligned} \min_{u \in \mathcal{U}, \delta \in \mathbb{R}^N} \quad & \|u\|^2 \\ \text{s.t.} \quad & L_f J_1(x) + L_g J_1(x) u \leq -\sigma_1(x) \\ & \vdots \\ & L_f J_N(x) + L_g J_N(x) u \leq -\sigma_N(x) \end{aligned}$$

We do not.

Sometimes they are compatible

$J_1 \rightarrow$ Learn to avoid circular region

$J_2 \rightarrow$ Learn to go to some point

<ADD “ANIMATION">

Sometimes they are compatible

$J_1 \rightarrow$ Learn to form a triangle

$J_2 \rightarrow$ Learn to send one robot to a point

<ADD “ANIMATION">

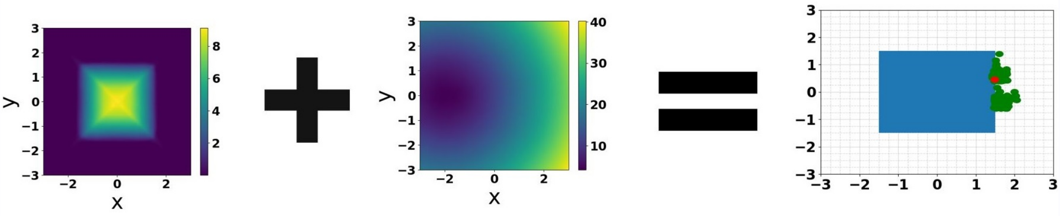
Sometimes they are NOT compatible

$J_1 \rightarrow$ Learn to avoid square-shaped region

$J_2 \rightarrow$ Learn to go to a point

<ADD “ANIMATION">

Sometimes they are NOT compatible



- The above are heat maps of the two value functions.
- Combining them results in the trajectory on the right.

Definitions of *Independence* and *Orthogonality*

$$
\begin{aligned}
\min_{u\in\mathcal{U},\delta\in\mathbb{R}^N} \quad & \|u\|^2 \\
\text{s.t.} \quad & L_f J_1(x) + L_g J_1(x) u \le -\sigma_1(x) \\
& \vdots \\
& L_f J_N(x) + L_g J_N(x) u \le -\sigma_N(x)
\end{aligned}
$$

Definitions of *Independence* and *Orthogonality*

$$J_1, \ldots J_N \text{ are } \textit{independent} \text{ at } x \in \mathcal{X} \Leftrightarrow L_g J_1(x)^\top, \ldots, L_g J_N(x)^\top \text{ are linearly independent}$$

$$J_1, \ldots J_N \text{ are } \textit{orthogonal} \text{ at } x \in \mathcal{X} \Leftrightarrow \langle L_g J_i(x)^\top, L_g J_j(x)^\top \rangle = 0 \ \forall \ i, j \in \{1, \ldots, N\}$$

Introducing "Interference" Input Cost

$$J_{N+1} \approx \min_{u(\cdot)} \int_t^{\infty} e^{-\beta\tau} \left(q_{N+1}(x) + \|u\|^2 + \sum_{i=1}^{N} (L_g J_i(x) u)^2 \lambda_i \right) d\tau$$

In Proposition 2, we show that by picking large enough values of λ_i and successfully fitting the cost functional, we can make the new task, J_{N+1}, independent to previously trained tasks $J_1, \dots, J_N$.

Variant of Value Iteration in Continuous Action Space

Can extend previous work in [2] to approximate this new cost functional.

$$J_{N+1} \approx \min_{u(\cdot)} \int_t^{\infty} e^{-\beta\tau} \left(q_{N+1}(x) + \|u\|^2 + \sum_{i=1}^{N} (L_g J_i(x) u)^2 \lambda_i \right) d\tau$$

At each iteration, use the following input to to estimate the next iteration of the value function.

$$u^{\star} = -\frac{1}{2} R(x)^{-1} (L_g J_{N+1}(x))^{\top}$$

$$R(x) = I + \sum_{i=1}^{N} (L_g J_i(x))^{\top} L_g J_i(x)$$

[2] M. Lutter, S. Mannor, J. Peters, D. Fox, and A. Garg, "Value Iteration in Continuous Actions, States and Time," in Proceedings of the 38th International Conference on Machine Learning, Jul. 2021, vol. 139, pp. 7224–7234.

Relating Orthogonality to the Optimal Input

Proposition 3:
Assume that $J_1, \ldots, J_N$ are independent at $x \in \mathcal{X}$.
If we train a new cost-to-go function, J_{N+1} to be independent to $J_1, \ldots, J_N$,
J_{N+1} is orthogonal to each of $J_1, \ldots, J_N$ at $x \in \mathcal{X}$.
$\Leftrightarrow$
the optimal input is $-\frac{1}{2} L_g J_{N+1}(x)^\top$.

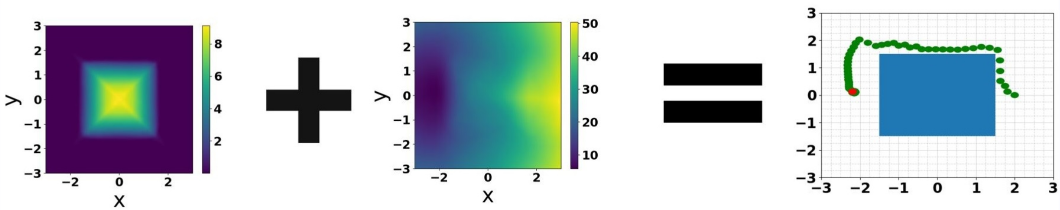
Now they ARE compatible.

$J_1 \rightarrow$ Learn to avoid square-shaped region

$J_2 \rightarrow$ Learn to go to a point

<ADD “ANIMATION">

Now they ARE compatible.



- The above are heat maps of the two value functions.
- Combining them results in the trajectory on the right.

We tried our idea on other scenarios.

Add pictures here.