Reliable Policy Iteration: Performance Robustness Across Architecture and Environment Perturbations

S.R. Eshwar[1] Aniruddha Mukherjee[2] **Kintan Saha**[1] Krishna Agarwal[1]
Gugan Thoppe[1] Aditya Gopalan[1] Gal Dalal[3]

[1]Indian Institute of Science, Bengaluru

[2]Kalinga Institute of Industrial Technology, Bhubaneswar

[3]NVIDIA Research, Israel

Indian Control Conference, 2025

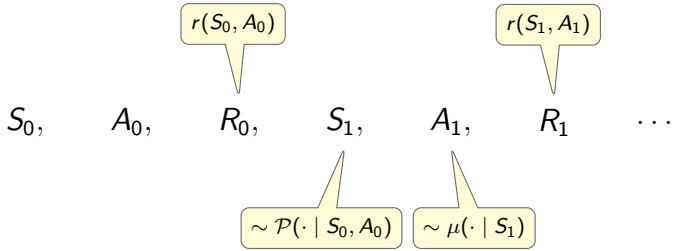
Overview

Markov Decision Process (MDP)

- States: $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$
- Actions: $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$
- State transition probabilities: $\mathcal{P}(s'|s, a)$
- Rewards: $r(s, a)$
- Reward discounting factor: $0 \leq \gamma < 1$

Reinforcement Learning (I)

- Agent interacts with an environment modeled as an MDP $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$
- Policy: A decision rule $\mu : \mathcal{S} \to \Delta(\mathcal{A})$ adopted by the agent to take an action at each state
- Trajectory (τ):



Reinforcement Learning (II)

- **Action–value function:**

$$Q_\mu(s,a) = \mathbb{E}_{\tau\sim\mu}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\,\middle|\, s_0 = s, a_0 = a\right]$$

Reinforcement Learning (II)

- **Action–value function:**

$$Q_\mu(s,a) = \mathbb{E}_{\tau\sim\mu}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\,\middle|\, s_0 = s, a_0 = a\right]$$

- **Bellman operator:**

$$(T_\mu Q)(s,a) = r(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a) \sum_{a'\in\mathcal{A}} \mu(a'|s')Q(s',a')$$

where $T_\mu Q_\mu = Q_\mu$

Reinforcement Learning (II)

- **Action–value function:**

$$Q_\mu(s,a) = \mathbb{E}_{\tau\sim\mu}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\,\middle|\, s_0=s, a_0=a\right]$$

- **Bellman operator:**

$$(T_\mu Q)(s,a) = r(s,a) + \gamma\sum_{s'\in\mathcal{S}} P(s'|s,a)\sum_{a'\in\mathcal{A}}\mu(a'|s')Q(s',a')$$

where $T_\mu Q_\mu = Q_\mu$

- **Bellman Optimality operator:**

$$(TQ)(s,a) = r(s,a) + \gamma\sum_{s'\in\mathcal{S}} P(s'|s,a)\max_{a'\in\mathcal{A}} Q(s',a')$$

where $Q^* = \max_\mu Q_\mu, \quad TQ^* = Q^*, \quad \mu^* \in \mathrm{Greedy}(Q^*)$

Overview

Policy Iteration

Classical Policy Iteration (Howard, '60)

- Initialize a policy μ_0
- For $k = 0, 1, 2, \ldots$
  - **Policy Evaluation:** Compute the exact action–value function

$$Q_{\mu_k} = T_{\mu_k} Q_{\mu_k}$$

  - **Policy Improvement:** Update the policy greedily

$$\mu_{k+1} \in \text{Greedy}(Q_{\mu_k})$$

  - Terminate when the Q-value stops changing

Policy Iteration: Guarantees

- **Monotonic Improvement**

$$Q_{\mu_0} \leq Q_{\mu_1} \leq Q_{\mu_2} \leq \dots$$

Policy Iteration: Guarantees

- **Monotonic Improvement**

$$Q_{\mu_0} \leq Q_{\mu_1} \leq Q_{\mu_2} \leq \dots$$

- **Finite Convergence**

$$Q_{\mu_k} \to Q^* \quad \text{in finitely many iterations}$$

Overview

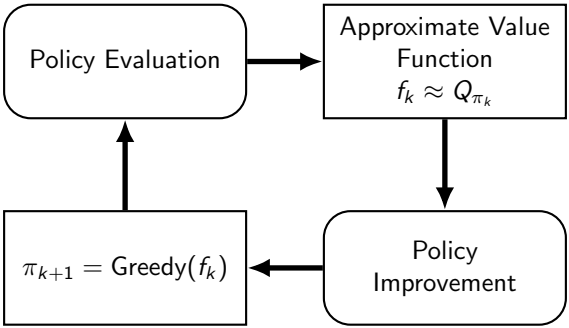
Policy Iteration: Challenges and Function Approximation

- Exact policy evaluation is infeasible in large-scale problems:
  - Robotics, control: continuous state–action spaces
  - Games, LLMs: combinatorially large spaces
- Must use **approximate policy evaluation**

$$Q_{\mu_k}(s,a) \approx f(s,a), \quad f \in \mathcal{F}$$

- $\mathcal{F}$ may be:
  - Linear function classes
  - Neural networks

Approximate Policy Iteration



But no monotonic improvement guarantees

"... approximate policy iteration methods are based on the idea of 'approximation in value space' and hence also on the hypothesis that a more accurate cost-to-go approximation will yield a better ... policy through the policy improvement equation. This is ... **by no means a self-evident hypothesis**, *and may in fact not even be true in a given problem ... the associated convergence behavior is complex, and involves potentially* **damaging oscillations**, *which despite their initial description long ago, are still not well understood at present."*

– Bertsekas, '11 *Approximate Policy Iteration: A Survey and Some New Methods*

Function Approximation - Oscillations in CartPole

Overview

Reliable Policy Iteration

Key Idea: Reformulate policy evaluation as Bellman-constrained optimization.

- Initialize policy μ_0 and value estimate $f_0 \in \mathcal{F}$
- For $k = 0, 1, 2, \ldots$
  - **Policy Evaluation:**

$$f_{k+1} \in \arg\max_{f \in \mathcal{F}} \|f - f_k\| \quad \text{s.t.} \quad T_{\mu_k} f \geq f \geq f_k$$

  - **Policy Improvement:**

$$\mu_{k+1} \in \text{Greedy}(f_{k+1})$$

RPI: Interpretation

$$\boxed{f_{k+1} \in \arg\max_{f \in \mathcal{F}} \|f - f_k\| \quad \text{s.t.} \quad T_{\mu_k} f \geq f \geq f_k}$$

- Constraint $T_{\mu_k} f \geq f$ ensures $Q_{\mu_k} \geq f_k$
- Constraint $f \geq f_k$ enforces monotonic improvement
- Objective *maximizes movement* away from f_k
- New estimate f_{k+1} is a **tight lower bound** of Q_{μ_k}

RPI: Guarantees

Generalizes PI: RPI matches PI under tabular setting

For *arbitrary function classes*, assuming $T_{\mu_0} f_0 \geq f_0$:

- **Feasibility:** Optimization problem remains feasible at all iterations
- **Monotonicity:**

$$f_0 \leq f_1 \leq f_2 \leq \dots \qquad \text{and} \qquad f_k \leq Q_{\mu_k}$$

RPI: Extension to Deep RL

- **Goal:** Approximate the Bellman-constrained evaluation $f_{k+1} = \arg\max_f \|f - f_k\|$ s.t. $T_{\mu_k} f \geq f \geq f_k$ using samples.

RPI: Extension to Deep RL

- **Goal:** Approximate the Bellman-constrained evaluation $f_{k+1} = \arg\max_f \|f - f_k\|$ s.t. $T_{\mu_k} f \geq f \geq f_k$ using samples.
- **Challenges:**
  - Constraints per (s, a) pair
  - Cannot compute T_{μ_k}, since we do not have access to $\mathcal{P}$

RPI: Extension to Deep RL

- **Goal:** Approximate the Bellman-constrained evaluation $f_{k+1} = \arg\max_f \|f - f_k\|$ s.t. $T_{\mu_k} f \geq f \geq f_k$ using samples.
- **Challenges:**
  - Constraints per (s, a) pair
  - Cannot compute T_{μ_k}, since we do not have access to $\mathcal{P}$
- **Approach:**
  - Convert the constrained optimization problem into unconstrained objective using penalty based method
  - Use samples to run stochastic gradient descent on the new unconstrained optimization problem

RPI: Critic Loss

$$\mathcal{L}_{\mathrm{RPI}}(f) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \Big(- c\, f(s_i, a_i) + \lambda_1 [f(s_i, a_i) - y_i]_+ + \lambda_2 [q_{\min} - f(s_i, a_i)]_+ \Big)$$

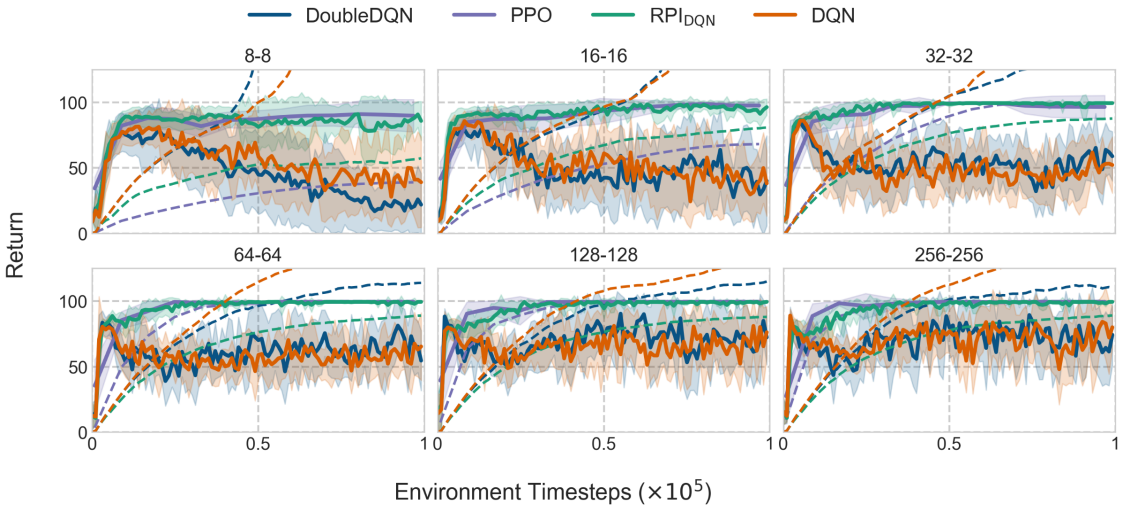
- $[x]_+ = \max(x, 0), \quad y_i \approx T_\mu f(s_i, a_i) = r_i + \gamma f(s_i', a_i')$ (target network).
- **Role of each term:**
  - $-c\, f$: encourages monotone value improvement
  - $[f - y_i]_+$: penalizes violation of $f \leq T_\mu f$
  - $[q_{\min} - f]_+$: enforces a lower bound ($f \geq f_k$)

Overview

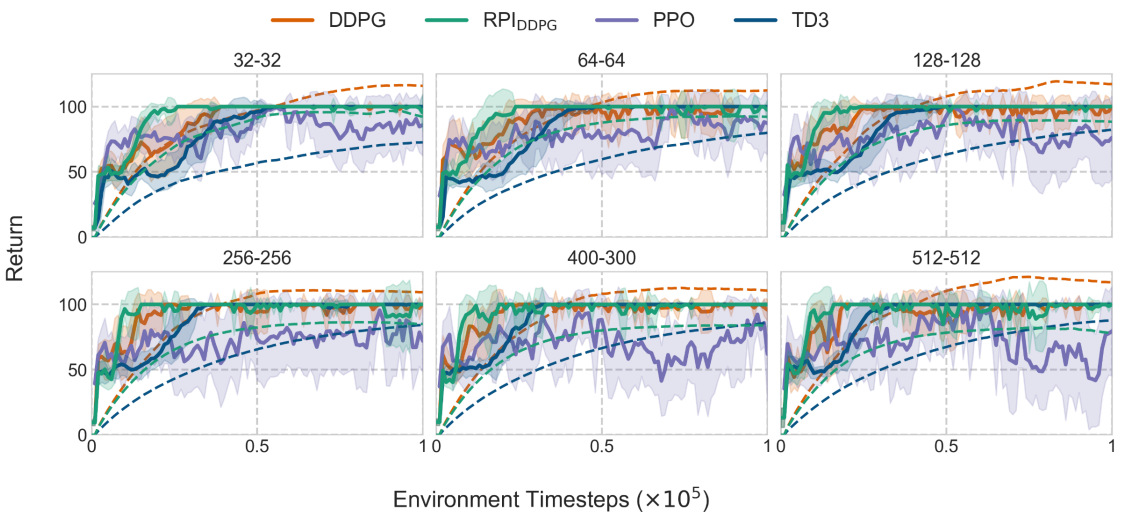
Experiments

- Verify RPI guarantees empirically:
  - Critic remains a lower bound to Monte Carlo returns i.e. $f_k \leq Q_{\mu_k}$
- Two experimental axes:
  1. **Function Approximation:** Vary the network size of the neural net approximating the Q-function
  2. **Environment Dynamics:** Modify mass, gravity, etc to test the robustness to perturbations in the environment.
- Environments chosen (OpenAI Gym):
  1. CartPole: Continuous state, finite action
  2. Inverted Pendulum: Continuous state, continuous action

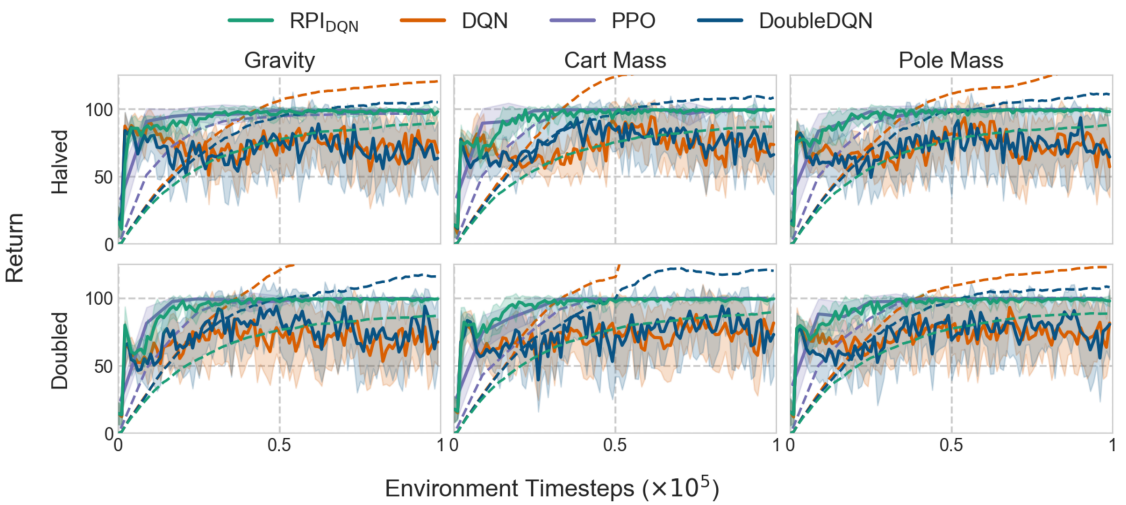
Function Approximation: CartPole



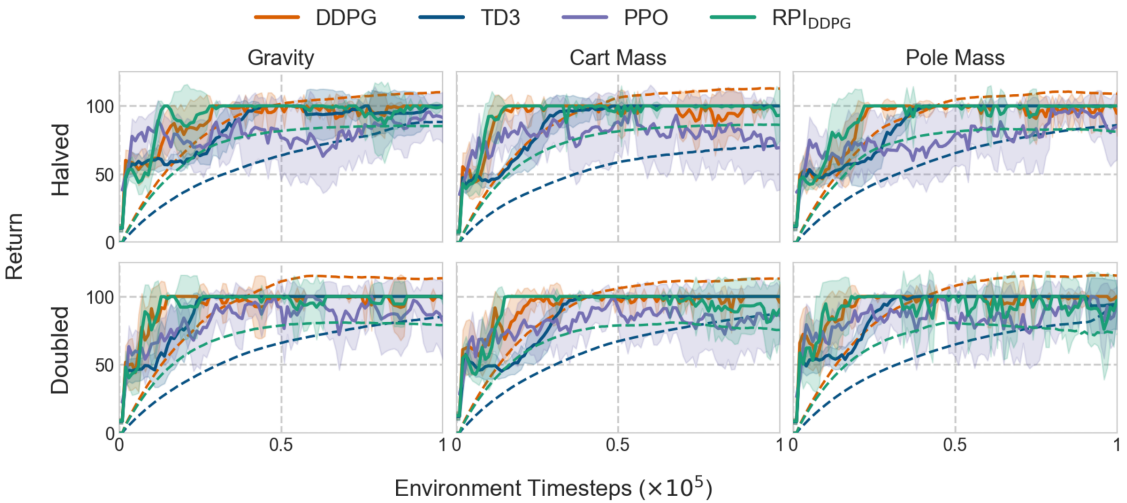
Function Approximation: Inverted Pendulum



Environment Dynamics: CartPole



Environment Dynamics: Inverted Pendulum



Future Directions

- Extending to more complex environments such as MuJuCo and Atari.

Thank You

Questions?