Meta Learning the Step Size in Policy Gradient Methods

Candidate: Francesco Corda
Advisors: Prof. Marcello Restelli, Eng. Luca Sabbioni

April 28, 2021
Master Thesis in Computer Science and Engineering

Outline

1 Reinforcement Learning

2 Meta Learning

Background Knowledge & State of the Art

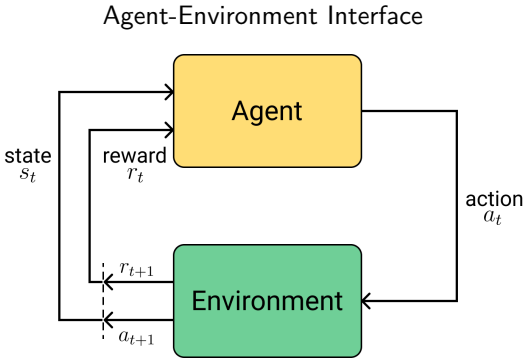
3 Proposed Solution

4 Experimental Results

5 Conclusions

Contributions & Open Questions

Sequential Decision Making Problems



Markov Decision Process:

- **MDP**: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma, \mu \rangle$
- **Return**: $G_{t_0} = \sum_{t=t_0}^{T} \gamma^{t-t_0} R(s_t, a_t)$
- **Parametric Policy**: $a \sim \pi_{\boldsymbol{\theta}}(\cdot \mid s)$
- **Action-Value Function**: $Q_\pi(s, a) = \mathop{\mathbb{E}}_{P,\pi} [G_0 \mid s_0 = s, a_0 = a]$
- **Goal**: $\arg\max_{\boldsymbol{\theta}} J_{\pi_{\boldsymbol{\theta}}} = \mathop{\mathbb{E}}_{P,\pi_{\boldsymbol{\theta}}} [G_0]$

Policy Gradient Methods

Policy Gradients (PG) main concepts:

- **Goal**: optimize policy $\pi_{\boldsymbol{\theta}}$ with a series of **Gradient Ascent** steps

Policy Gradient Update

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{\alpha} \nabla J(\boldsymbol{\theta}_t)$$

$$\nabla J(\boldsymbol{\theta}) = \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} Q_\pi(s,a) \nabla \pi_{\boldsymbol{\theta}}(a \mid s) \qquad \text{(PGT)}$$

- **Hyperparameter**: step size $\boldsymbol{\alpha}$, regulates the magnitude of the update

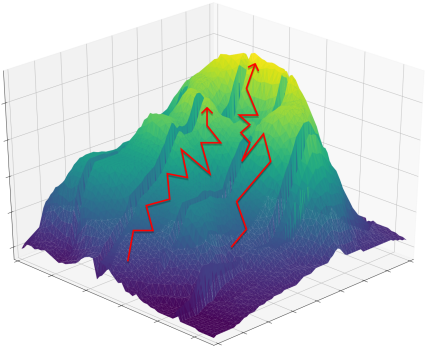
PG Methods - Drawbacks and Proposed Solution

Drawbacks:

- slow or **unstable convergence**
- high **sample inefficiency**
- manual **hyperparameters tuning**

Solution: learn a **Dynamic Step Size**

- maximize **learning speed**
- minimize **instability**



PG Gradient Ascent Example

Value Based Methods - Fitted Q Iteration

Main concepts:

- **Value-Based** methods: learn the optimal value function Q^*
- **Fitted Q Iteration** (FQI): approximate the Q function with a series of regressions

Main steps of FQI (Ernst et al., 2005):

1. Input set $\mathcal{F} = \{(s_i, a_i, s'_i, r_i)\}$
2. Build $\mathcal{TS} = \left\{\left(i^l, o^l\right)\right\}_l$, where: $i^l = \left(s_t^l, a_t^l\right)$ and $o^l = \hat{Q}_{n-1}\left(s_t^l, a_t^l\right)$
3. Use a **regression algorithm** to induce the function $\hat{Q}_n(\mathcal{S}, \mathcal{A})$

repeat

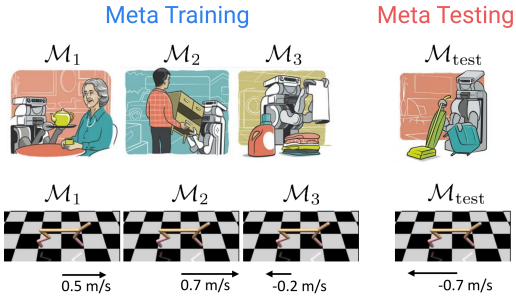
Meta Learning - Learning How to Learn

Main Goals:

- **learn quickly**, using few training samples
- **adapt to changes** and dynamic scenarios

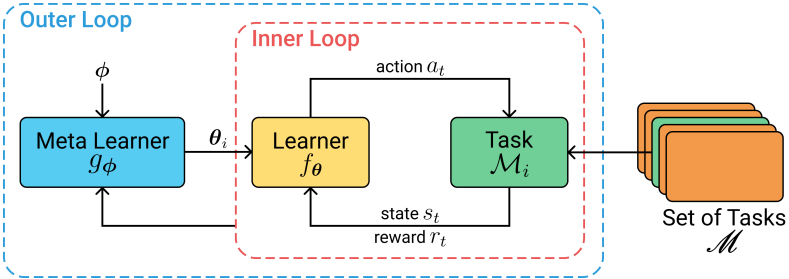
Meta Learning works **above Machine Learning**:

- the training samples are **learning instances**



Examples of Meta Learning Tasks in RL

Meta Reinforcement Learning

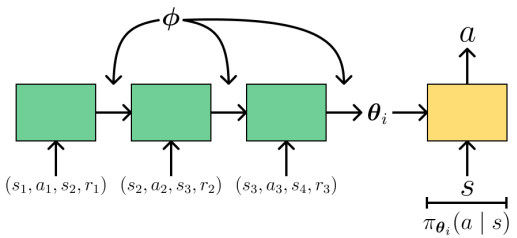


Meta-RL Two Levels Architecture

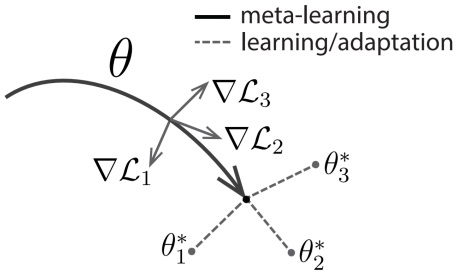
General Meta Reinforcement Learning (Meta-RL) training process:

1. Sample a **random MDP** $\mathcal{M}_i \sim \mathscr{M}$
2. Reset the hidden state of the **learner** f_θ
3. Optimize the model to the task $\mathcal{M}_i$ according to the **meta learner** g_ϕ

Meta-RL State of the Art



RL^2 (Duan et al., 2016),
Learning to RL (Wang et al., 2017)



MAML (Finn et al., 2017),
Reptile (Nichol et al., 2018)

Proposed Framework: Meta-MDP

Given a set of MDPs $\mathscr{M} = \{\mathcal{M}_\omega\}$, defined by a **context** $\omega \in \Omega$:

A **meta-MDP** is a tuple $\langle \mathcal{X}, \mathcal{H}, \mathcal{L}, \tilde{\gamma}, (\mathscr{M}, p), (\boldsymbol{\theta}, q), f \rangle$, where:

- $\mathcal{X}$ is the **meta state space**
- $\mathcal{H}$ is the **meta learning action**
- $\mathcal{L}$ is the **meta reward function**
- $f : \boldsymbol{\theta} \times \mathcal{H} \to \boldsymbol{\theta}$ is the update algorithm

Meta Reward Function Proposed:

$$\mathcal{L}(x, \boldsymbol{\theta}, h) := J_i(f(\boldsymbol{\theta}, h)) - J_i(\boldsymbol{\theta}) = \underbrace{J_i\left(\boldsymbol{\theta} + h \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta})\right) - J_i(\boldsymbol{\theta})}_{\text{Gradient Ascent Case}}$$

Lipschitz Meta-MDP

Lipschitz Continuity (LC)

$$\forall x, x' \in X, d_Y\left(f(x), f\left(x'\right)\right) \leqslant L_f d_X\left(x, x'\right)$$

Given a set of LC tasks $\mathscr{M} = \{\mathcal{M}_\omega\}_{\omega \in \Omega}$:

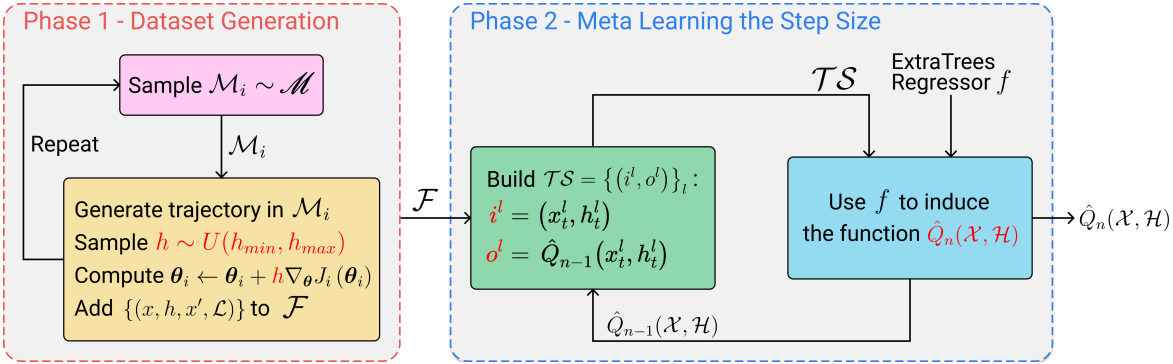
Bound in the Action-Value Function Q^π_ω

$$\left|Q^\pi_{\omega_1}(s,a) - Q^\pi_{\omega_2}(s,a)\right| \leqslant L_{\omega_Q}(\pi) d_\Omega(\omega_1, \omega_2),$$

where $L_{\omega_Q}(\pi) = \dfrac{L_{\omega_r} + \gamma L_{\omega_p} L_{V^\pi}(\omega)}{1-\gamma}$

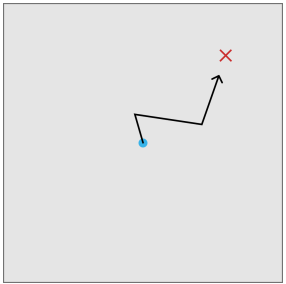
Similar guarantees have been derived for the gradient $\nabla J_\omega(\boldsymbol{\theta})$.

Proposed Algorithm: Learning the Step Size

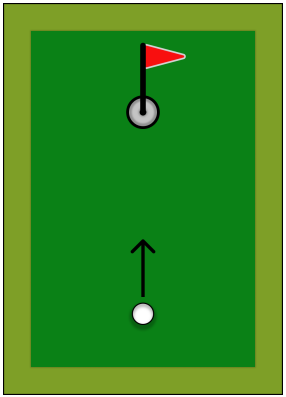


Meta Learning the Step Size with FQI:
Dataset Generation & Meta Training

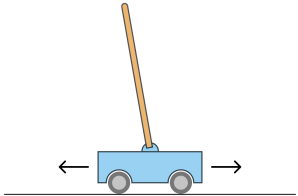
Experimental Settings



Navigation2D:
random goal point

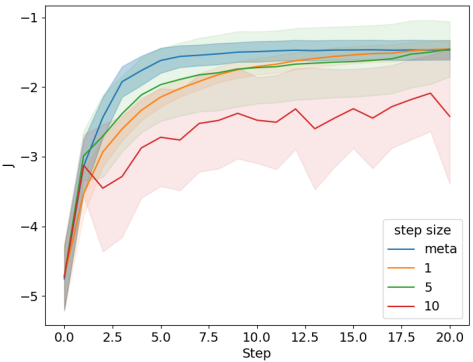


MiniGolf:
random putter length
and friction coefficient

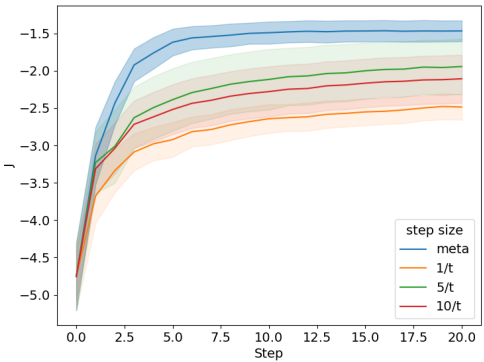


CartPole:
random gravity acceleration
and pole mass

Navigation2D - Dynamic Step Size vs Benchmarks

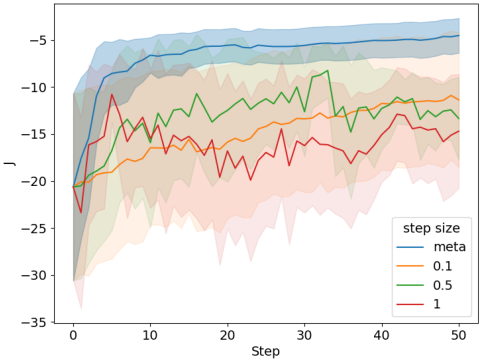


Nav2D - Expected Return J:
Dynamic Step Size vs Fixed Step Size α

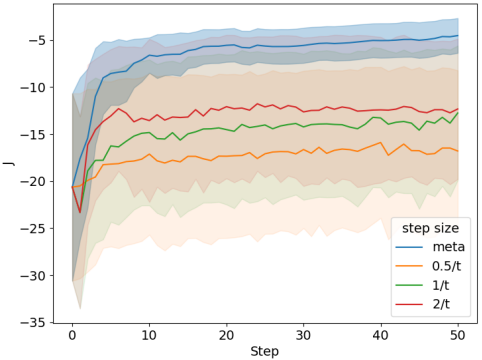


Nav2D - Expected Return J:
Dynamic Step Size vs Decreasing Step Size $\frac{\alpha}{t}$

MiniGolf - Dynamic Step Size vs Benchmarks

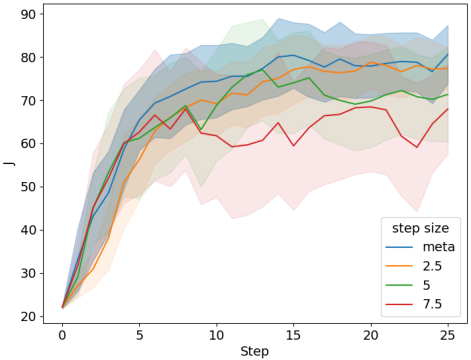


MiniGolf - Expected Return J:
Dynamic Step Size vs Fixed Step Size α

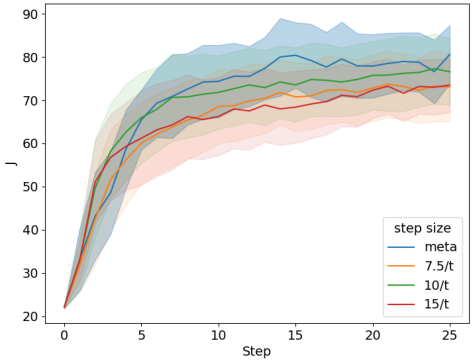


MiniGolf - Expected Return J:
Dynamic Step Size vs Decreasing Step Size $\frac{\alpha}{t}$

CartPole - Dynamic Step Size vs Benchmarks



CartPole - Expected Return J:
Dynamic Step Size vs Fixed Step Size α



CartPole - Expected Return J:
Dynamic Step Size vs Decreasing Step Size $\frac{\alpha}{t}$

Conclusions

Results achieved:

1. **Meta-MDP**: introduced new framework to represent meta-RL tasks;
2. **LC meta-MDP**: derived guarantees on the performance under the LC property;
3. **Dynamic Step Size**: proposed algorithm to learn a dynamic step size in meta-MDPs with PG learners;
4. **Experiments**: evaluated performance in different simulated settings.

Thank You for Your Attention

References

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Found. Trends Robot*, 2(1–2):1–142, August 2013. ISSN 1935-8253. doi: 10.1561/2300000021. URL https://doi.org/10.1561/2300000021.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning, 2016.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(18):503–556, 2005. URL http://jmlr.org/papers/v6/ernst05a.html.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

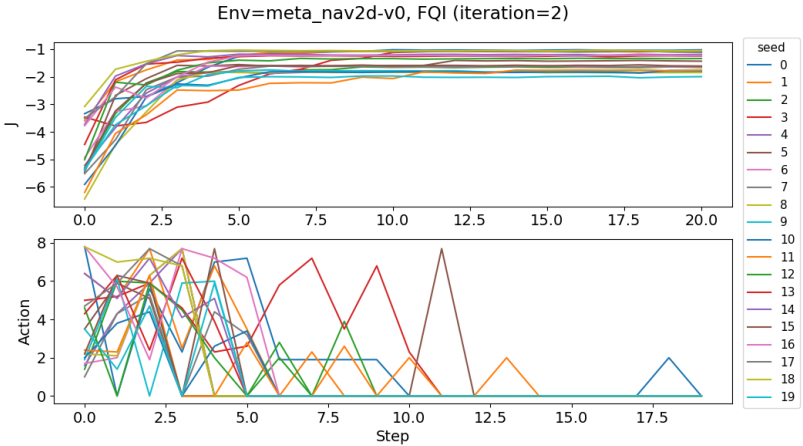
Ke Li and Jitendra Malik. Learning to optimize, 2016.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms, 2018.

Matteo Pirotta, Marcello Restelli, and Luca Bascetta. Policy gradient in lipschitz markov decision processes. *Machine Learning*, 100(2-3):255–283, 2015.

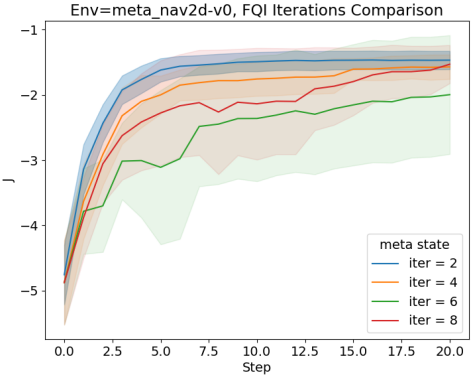
Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn, 2017.

Navigation2D Details

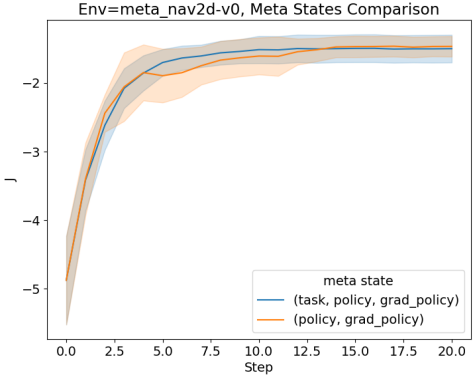


Dynamic PG - Expected return J and step size α at each meta step

Navigation2D Details

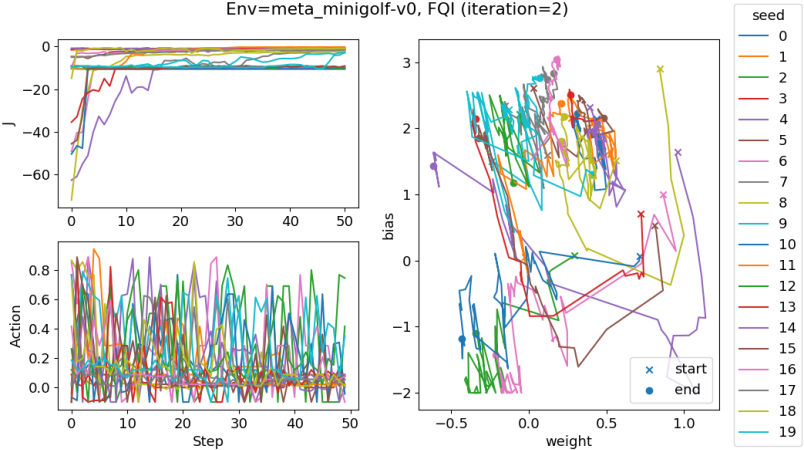


Performance of different iterations of the FQI algorithm



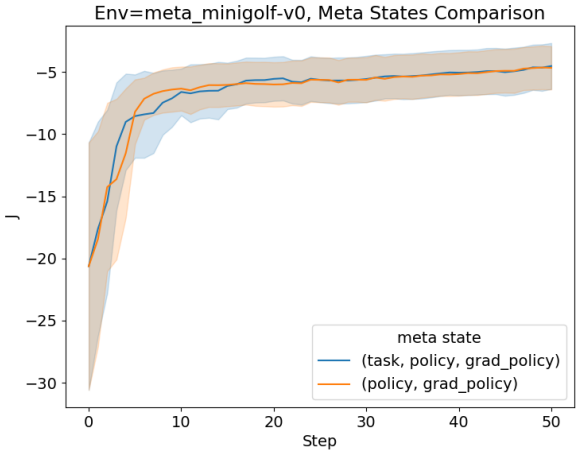
Performance of two meta states:
$x_1 = \langle \boldsymbol{\theta}_t, \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}_t), \phi_{\mathcal{M}}(\mathcal{M}_t) \rangle$,
$x_2 = \langle \boldsymbol{\theta}_t, \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}_t) \rangle$

MiniGolf Details



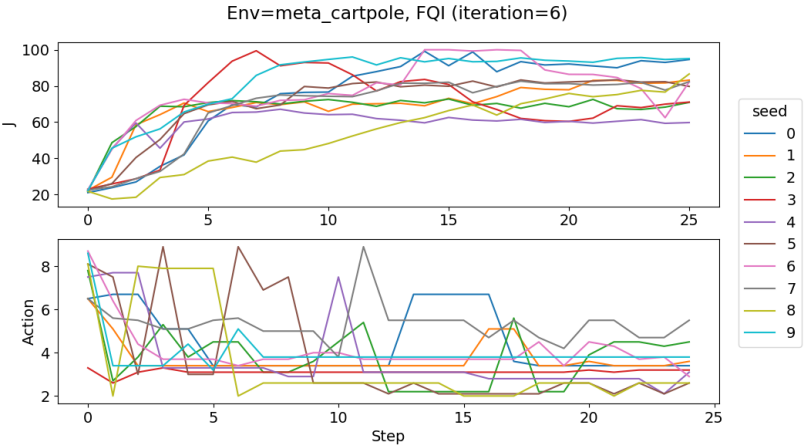
Dynamic PG - Expected return J, step size α and policy parameters $\boldsymbol{\theta}$ at each meta step

MiniGolf Details



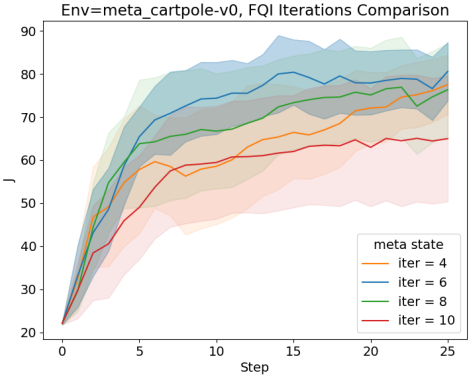
Performance of two meta states: $x_1 = \langle \boldsymbol{\theta}_t, \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}_t), \phi_{\mathcal{M}}(\mathcal{M}_t) \rangle$, $x_2 = \langle \boldsymbol{\theta}_t, \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}_t) \rangle$

CartPole Details

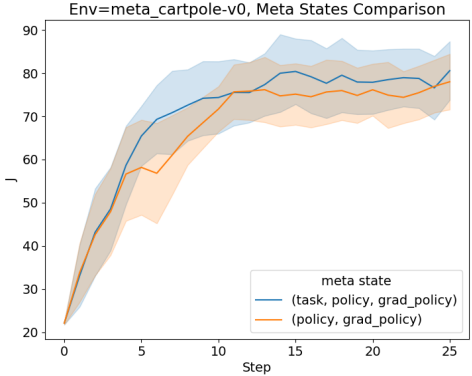


Dynamic PG - Expected return J and step size α at each meta step

CartPole Details



Performance of different iterations of the FQI algorithm



Performance of two meta states:
$x_1 = \langle \boldsymbol{\theta}_t, \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}_t), \phi_{\mathcal{M}}(\mathcal{M}_t) \rangle$,
$x_2 = \langle \boldsymbol{\theta}_t, \nabla_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}_t) \rangle$