Trustworthy AI Autonomy

M3-1: Safe Reinforcement Learning

Ding Zhao

Assistant Professor

Carnegie Mellon University

Carnegie Mellon University

Safe AI Lab @CMU

Plan for today

- Safe RL methods
  - Safety Gym environment
  - Trust-region search: TRPO, PPO
  - Constrained MDP

OpenAI Safety Gym

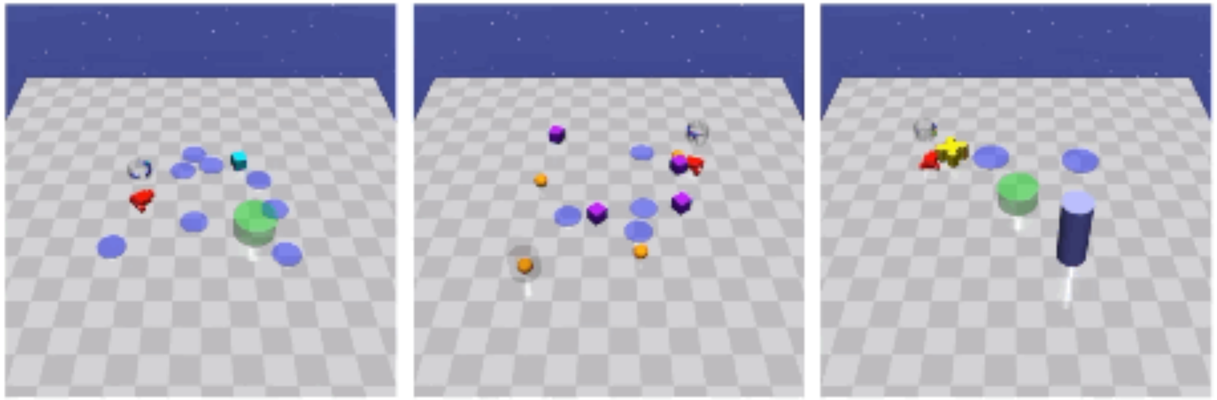
Benchmarking Safe Exploration in Deep Reinforcement Learning

Alex Ray* OpenAI

Joshua Achiam* OpenAI

Dario Amodei OpenAI

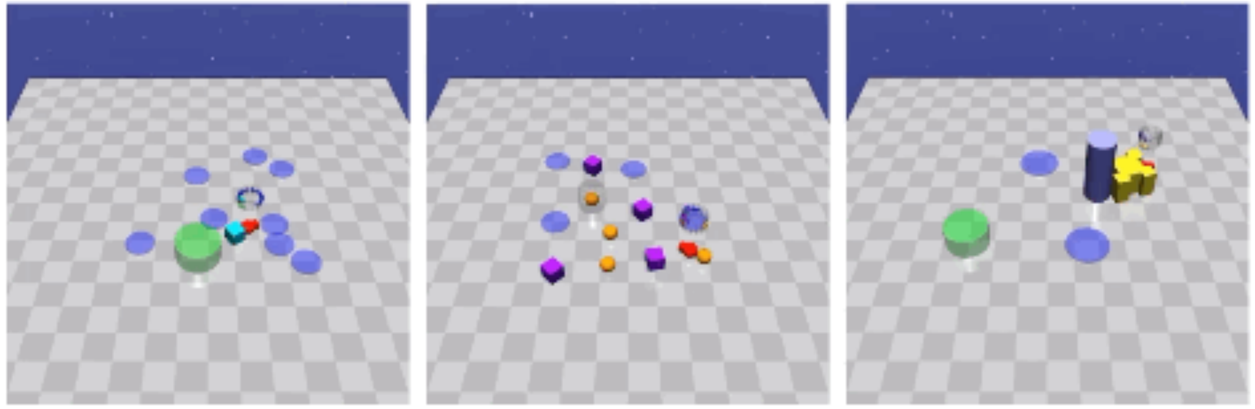
Car has two independently-driven parallel wheels and a free-rolling rear wheel. For this robot, turning and moving forward or backward require coordinating both of the actuators.



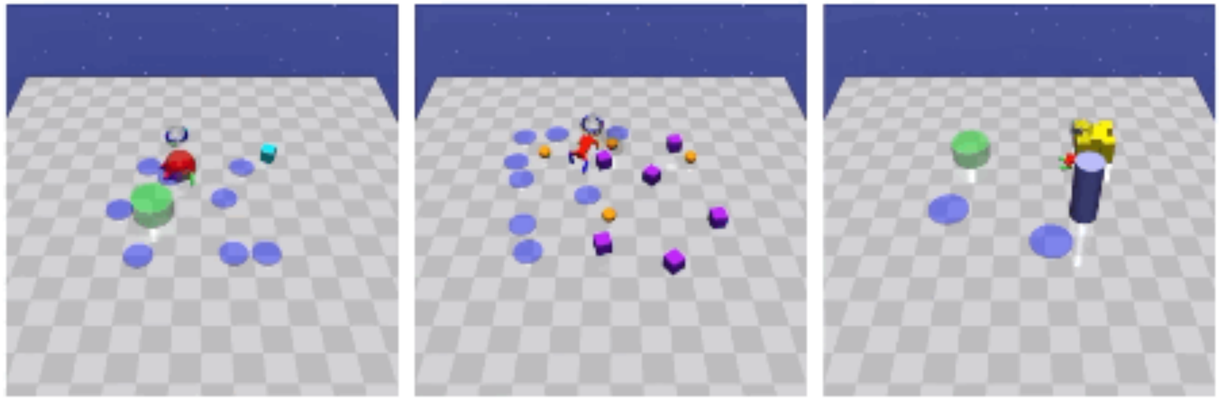
Goal: Move to a series of goal positions. **Button:** Press a series of goal buttons. **Push:** Move a box to a series of goal positions.

Point is a simple robot constrained to the 2D plane, with one actuator for turning and another for moving forward or backward. Point has a front-facing small square which helps with the Push task.

Doggo is a quadruped with bilateral symmetry. Each of its four legs has two controls at the hip, for azimuth and elevation relative to the torso, and one in the knee, controlling angle. A uniform random policy keeps the robot from falling over and generates travel.



Goal: Move to a series of goal positions. **Button:** Press a series of goal buttons. **Push:** Move a box to a series of goal positions.

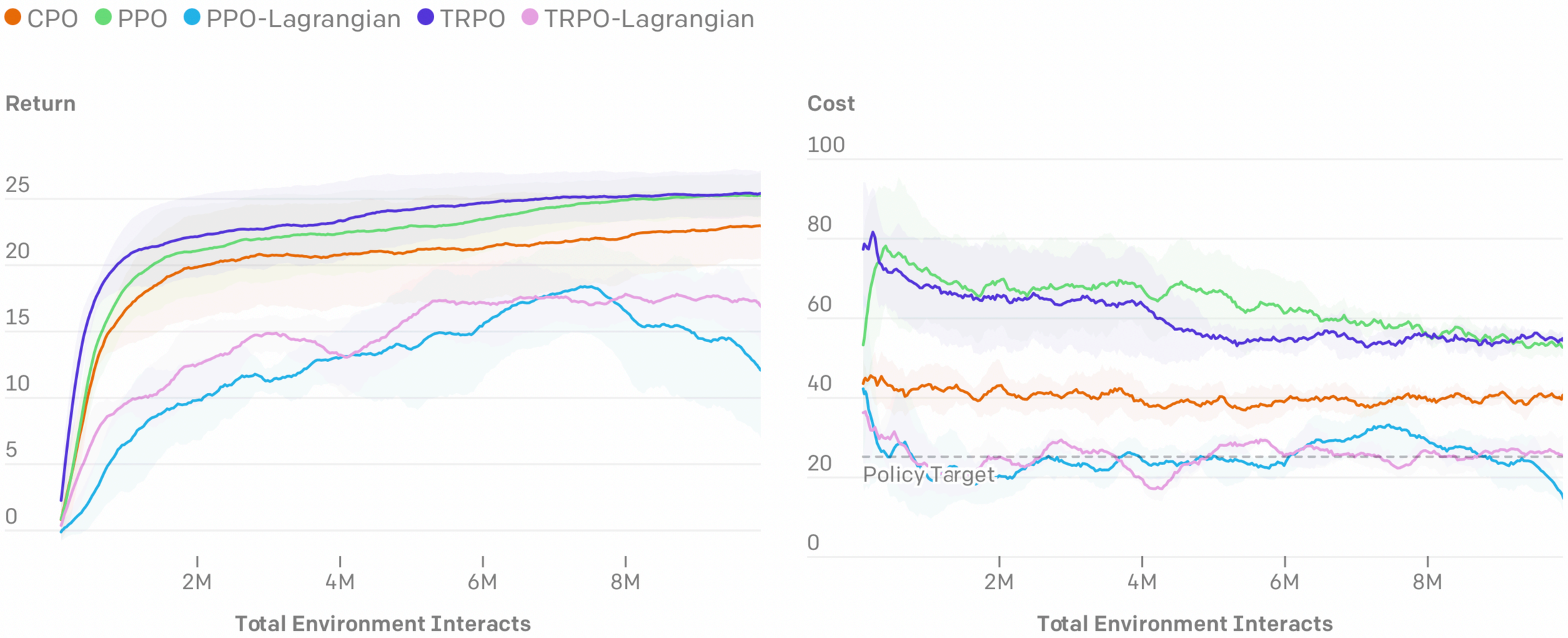


Goal: Move to a series of goal positions. **Button:** Press a series of goal buttons. **Push:** Move a box to a series of goal positions.

https://openai.com/blog/safety-gym/

Return and cost trade off against each other meaningfully

Robot Point Task Goal Level 1



Safe Reinforcement Learning

- *“Safe Reinforcement Learning can be defined as the process of learning policies that maximize the expectation of the return in problems in which it is important to ensure reasonable system performance and/or respect safety* ***constraints*** *during the learning and/or deployment processes.”*

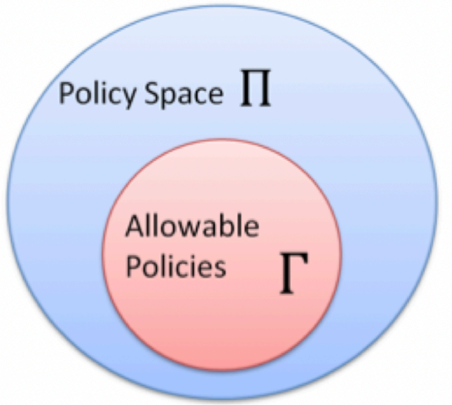
- Where to add the constraints?
  - Add constraints to the exploration procedure: TRPO, PPO
  - Add constraints to the optimization criteria (performance/constraints): CMDP

García, Javier, and Fernando Fernández. "A comprehensive survey on safe reinforcement learning." Journal of Machine Learning Research 16.1 (2015): 1437-1480.

Constrained Markov Decision Process

- Augment the MDP with a set C of auxiliary cost functions, $C_1, \ldots, C_m$ (with each one a function $C_i : S \times A \times S \to \mathbb{R}$ mapping transition tuples to costs, like the usual reward), and limits $d_1, \ldots, d_m$. Let $J_{C_i}(\pi)$ denote the expected discounted return of policy π with respect to cost function

$$C_i : J_{C_i}(\pi) = \mathop{\mathrm{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t C_i \left(s_t, a_t, s_{t+1} \right) \right].$$



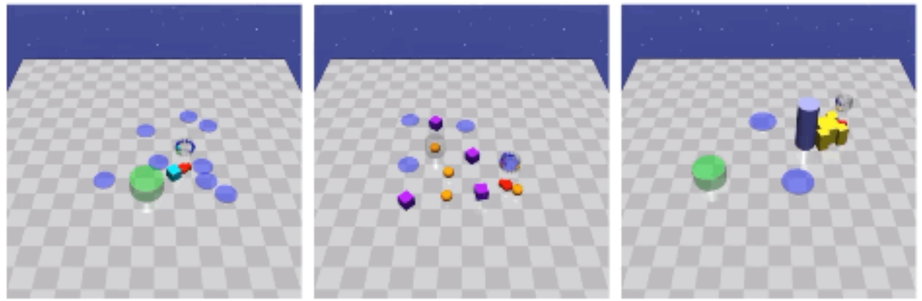
- The set of feasible stationary policies for a CMDP is then

$$\Pi_C \doteq \left\{ \pi \in \Pi : \forall i, J_{C_i}(\pi) \leq d_i \right\}$$

- and the reinforcement learning problem in a CMDP is

$$\pi^* = \arg\max_{\pi \in \Pi_C} J(\pi)$$

Point is a simple robot constrained to the 2D plane, with one actuator for turning and another for moving forward or backward. Point has a front-facing small square which helps with the Push task.



Goal: Move to a series of goal positions. **Button:** Press a series of goal buttons. **Push:** Move a box to a series of goal positions.

Safe RL methods

- Constrained Markov Decision Process
- Lagrangian TRPO and PPO
  - Trust Region Policy Optimization (TRPO)
  - Proximal Policy Optimization (PPO)
- Constrained Optimization-based Methods

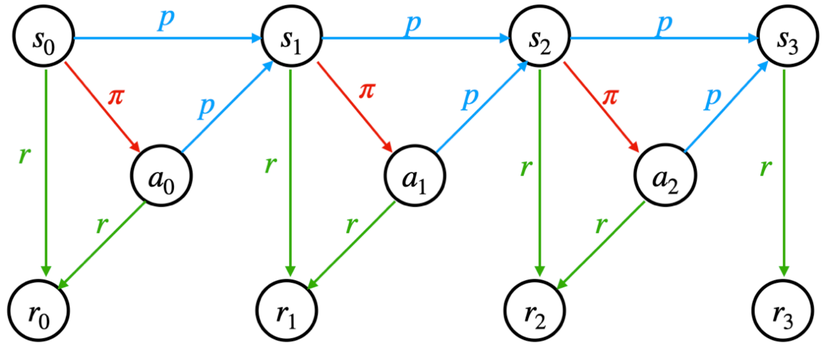
I will focus on the key motivations and intuitions. For details of implementation, you can read these blogs and corresponding papers.

https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

Recap: Markov Decision Process

- Defined by: $(\mathcal{S}, \mathcal{A}, r, p)$
  - $\mathcal{S}$: set of possible states
  - $\mathcal{A}$: set of possible actions
  - r : reward function
  - p : dynamics function

$$s_{t+1} \sim p\left(\cdot \mid s_t, a_t\right)$$
$$a_t \sim \pi\left(\cdot \mid s_t\right)$$
$$r_t \sim r\left(\cdot \mid s_t, a_t\right)$$



- Goal of MDP: given $(\mathcal{S}, \mathcal{A}, r, p)$, $\rho_0(\cdot)$, T or γ, we want

$$\pi^* = \arg\max_\pi \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Baselines for Safe RL algorithms

- Trust Region Policy Optimization (TRPO)
- Proximal Policy Optimization (PPO)

I will focus on the key motivations and intuitions. For details of implementation, you can read these blogs and corresponding papers.

https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

Trust Region Policy Optimization (TRPO)

- Safety issue with Vanilla Policy Gradient (VPG)
- $J(\theta, \mathscr{D}_{\pi_\theta}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,|\, \pi_\theta\right], \quad \theta^* = \arg\max_\theta J(\theta, \mathscr{D}_{\pi_\theta}), \quad \theta_{i+1} = \theta_i + \alpha \nabla_\theta J(\theta)\,|_{\theta=\theta_i}, \quad \theta_i \to \theta^*$
- We control the learning rate in parameter space, but due to the nonlinearity of neural networks, even a small change in parameter may lead to very different policies (output of the NN)—so a single bad step can collapse the training procedure. This makes it dangerous to use large step sizes with VPG, thus hurting its sample efficiency and induce risks.
- TRPO and a simpler algorithm PPO were invented to resolve this issue by defining the update constraint not on parameter space but directly on policy space.

Trust Region Policy Optimization (TRPO)

- Let π_θ denote a policy with parameters θ. We still have the similar goal with an additional constraint

$$\theta_{k+1} = \arg\max_\theta L\left(\theta_k, \theta\right)$$

$$\text{s.t.} \quad \bar{D}_{KL}\left(\theta \| \theta_k\right) \leq \delta$$

- where $L\left(\theta_k, \theta\right)$ is called the surrogate advantage, we use the old parameters to evaluate each action, and change the policy to increase the chance of "good" actions and decrease the chance of "bad" ones

$$L\left(\theta_k, \theta\right) = \mathbb{E}_{s,a\sim\pi_{\theta_k}}\left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)} A^{\pi_{\theta_k}}(s,a)\right]$$

- $\bar{D}_{KL}\left(\theta \| \theta_k\right)$ is an average KL-divergence between policies across states visited by the old policy. In other words, when parameters change, the chance of performance an action should not change much

$$\bar{D}_{KL}\left(\theta \| \theta_k\right) = \mathop{\mathbb{E}}_{s\sim\pi_{\theta_k}}\left[D_{KL}\left(\pi_\theta(\cdot \mid s) \| \pi_{\theta_k}(\cdot \mid s)\right)\right]$$

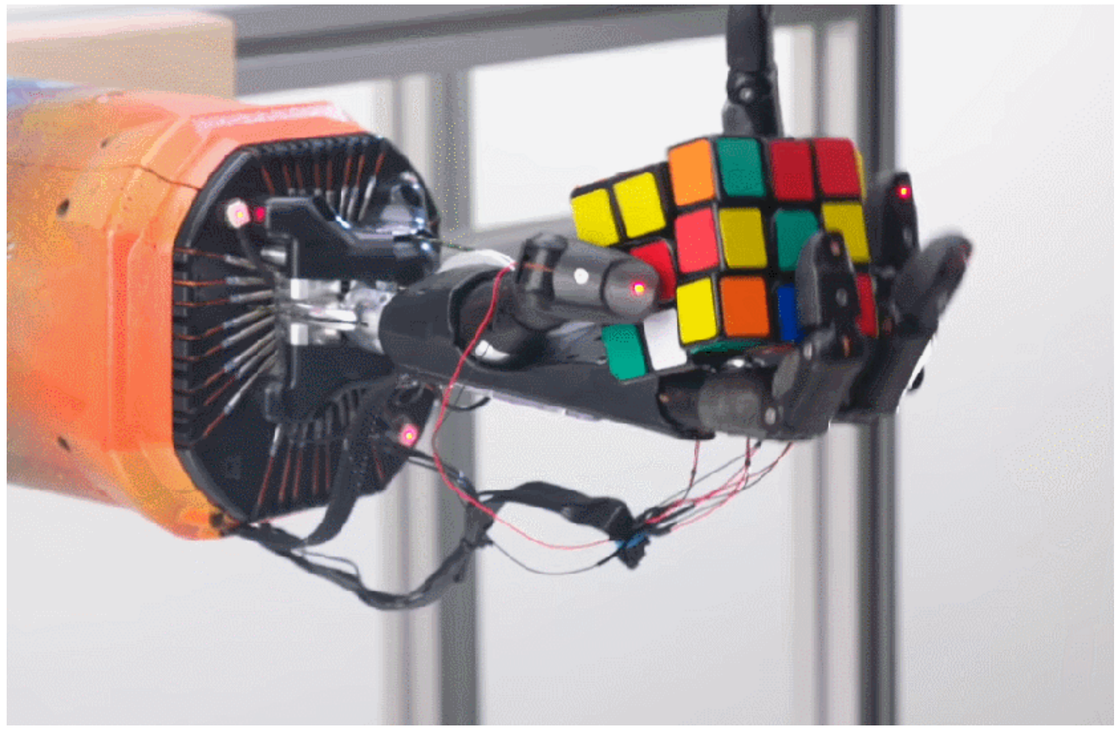
$$\text{Kullback–Leibler divergence} \quad D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$

Proximal Policy Optimization (PPO)

- PPO is motivated by the same question as TRPO: how can we take the biggest possible improvement step on a policy using the data we currently have, without stepping so far that we accidentally cause performance collapse?
- Where TRPO tries to solve this problem with a complex K-L divergence (second order) method, PPO is a family of first-order methods that use a few other tricks to keep new policies close to old.
- PPO methods are significantly simpler to implement, and empirically seem to perform at least as well as TRPO.
- PPO has been widely used in practice for its fast speed and simple implementation

https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

Famous tests using PPO



PPO algorithm

- Still the same goal

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}_{s,a\sim\pi_{\theta_k}} \left[L\left(s, a, \theta_k, \theta\right) \right]$$

- But incorporate the constraint into the objective function with a simple clip g

$$L\left(s, a, \theta_k, \theta\right) = \min\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)} A^{\pi_{\theta_k}}(s, a), \quad g_\epsilon\left(A^{\pi_{\theta_k}}(s, a)\right) \right)$$

$$\text{where } g_\epsilon(A) = \begin{cases} (1+\epsilon)A & A \geq 0 \quad \text{(good action)} \\ (1-\epsilon)A & A < 0 \quad \text{(bad action)} \end{cases}$$

- For good actions, π_θ increase with a ceiling at $(1+\epsilon)\pi_{\theta_k}$
- For good actions, π_θ decreases with a floor at $(1-\epsilon)\pi_{\theta_k}$

Different ways to add constraints

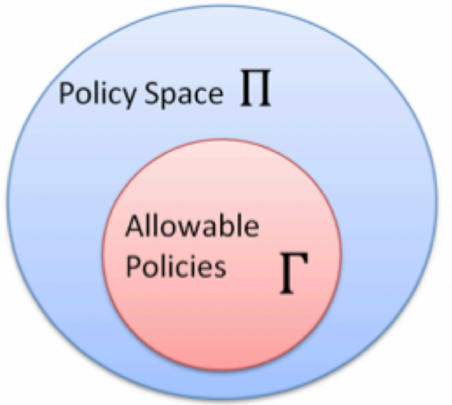
- VPG
  - $\theta_{i+1} = \theta_i + \alpha \nabla_\theta J(\theta)\,|_{\theta=\theta_i}$
  - Use a learning rate to constrain the change of θ in π_θ
- TRPO
  - $\bar{D}_{KL}\left(\theta||\theta_k\right) = \underset{s\sim\pi_{\theta_k}}{\mathbb{E}}\left[D_{KL}\left(\pi_\theta(\cdot\mid s)\|\pi_{\theta_k}(\cdot\mid s)\right)\right]$
  - Use KL divergence to bound the change of the policy distribution π_θ from π_{θ_k}. We may still have a large spike of change sometimes.
- PPO
  - $L\left(s,a,\theta_k,\theta\right) = \min\left(\frac{\pi_\theta(a\mid s)}{\pi_{\theta_k}(a\mid s)}A^{\pi_{\theta_k}}(s,a),\quad g_\epsilon\left(A^{\pi_{\theta_k}}(s,a)\right)\right), g_\epsilon(A) = \begin{cases}(1+\epsilon)A & A\geq 0\\(1-\epsilon)A & A<0\end{cases}$
  - Bound the averaged ratio of π_θ and π_{θ_k} for each action

They are all on-policy approach.

Constrained Markov Decision Process

- Augment the MDP with a set C of auxiliary cost functions, $C_1, \ldots, C_m$ (with each one a function $C_i : S \times A \times S \to \mathbb{R}$ mapping transition tuples to costs, like the usual reward), and limits $d_1, \ldots, d_m$. Let $J_{C_i}(\pi)$ denote the expected discounted return of policy π with respect to cost function

$$C_i : J_{C_i}(\pi) = \mathop{\mathrm{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t C_i \left(s_t, a_t, s_{t+1} \right) \right].$$



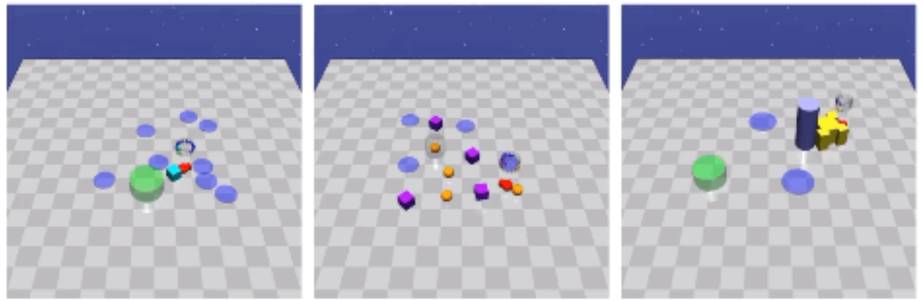
- The set of feasible stationary policies for a CMDP is then

$$\Pi_C \doteq \left\{ \pi \in \Pi : \forall i, J_{C_i}(\pi) \leq d_i \right\}$$

- and the reinforcement learning problem in a CMDP is

$$\pi^* = \arg\max_{\pi \in \Pi_C} J(\pi)$$

Point is a simple robot constrained to the 2D plane, with one actuator for turning and another for moving forward or backward. Point has a front-facing small square which helps with the Push task.



Goal: Move to a series of goal positions. **Button:** Press a series of goal buttons. **Push:** Move a box to a series of goal positions.

Primal-dual method 1.0

- Recall the constrained objective function in CMDP with one constraint:

$$\pi^* = \arg\max_\pi J(\pi)$$ maximizing the reward return

$$s.t. \quad J_C(\pi) \leq d$$ satisfying the constraint

- It has an equivalent Lagrangian formulation:

$$\pi^*, \lambda^* = \arg\min_{\lambda \geq 0} \max_\pi \quad L(\pi, \lambda)$$

$$L(\pi, \lambda) = J(\pi) - \lambda(J_C(\pi) - d)$$

- Dual problem: the outer minimization over the dual variable $\lambda \in \mathbb{R}_{\geq 0}$

Primal-dual method 1.0

- The Lagrangian formulation: $\pi^*, \lambda^* = \arg\min_{\lambda \geq 0} \max_{\pi} \quad L(\pi, \lambda)$

$$L(\pi, \lambda) = J(\pi) - \lambda(J_C(\pi) - d)$$

- Primal-dual (Lagrangian) algorithm:

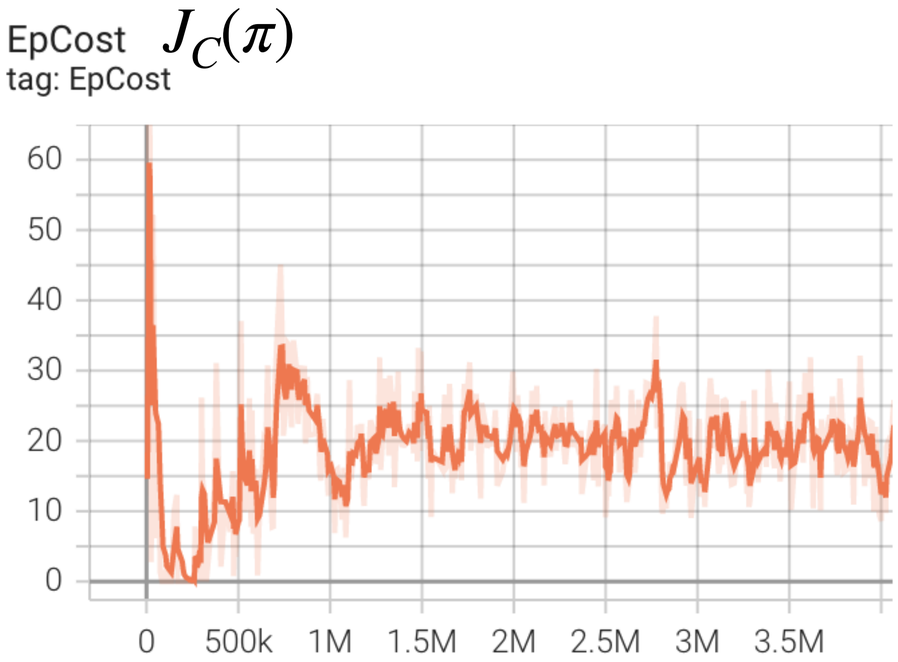
1. Fix λ, optimize π: $\pi_{i+1} = \arg\max_{\pi} \quad L(\pi_i, \lambda_i)$
2. Fix π, optimize λ: $\lambda_{i+1} = \arg\min_{\lambda \geq 0} \quad L(\pi_{i+1}, \lambda_i)$

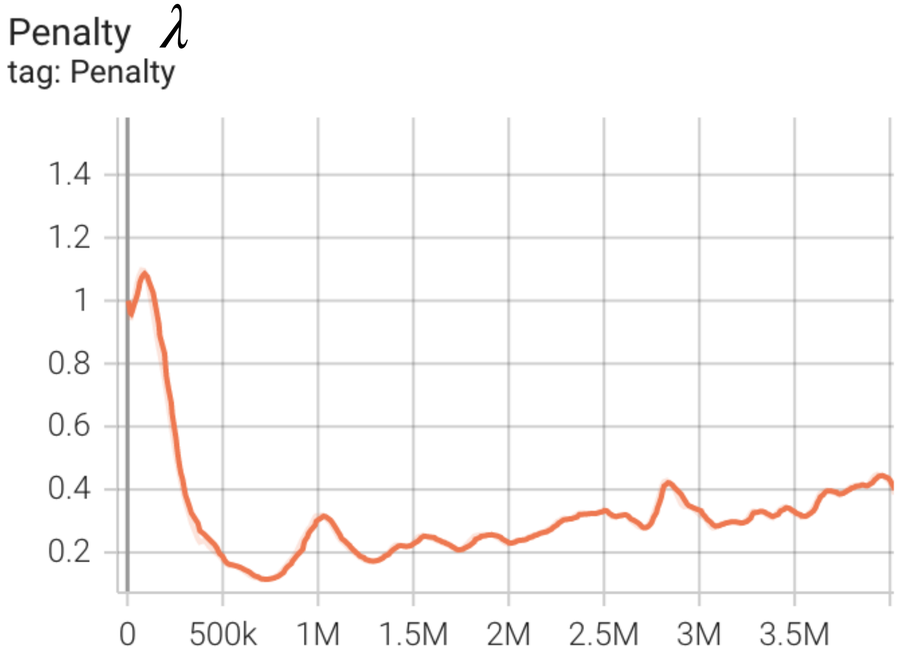
- Phase 1) can be done via any policy gradient-based RL algorithms
- Phase 2) is usually done via one-step gradient descend

$$\lambda_{i+1} = \max\left(0, \lambda_i - \eta \nabla_\lambda L(\pi_{i+1}, \lambda_i)\right) = \max\left(0, \lambda_i + \eta(J_C(\pi_{i+1}) - d)\right)$$

Primal-dual method 1.0

- Example: using PPO to optimize $\pi_{i+1} = \arg\max_{\pi} \quad J(\pi) - \lambda(J_C(\pi) - d)$
- PPO-Lagrangian algorithm





Primal-dual method 1.0

- What's the problem?
    - The dual problem is only approximately solved: $\lambda \to 0$ when $J_C(\pi)$ satisfies the constraint, $\lambda \to +\infty$ we fail

$$\pi^*, \lambda^* = \arg\min_{\lambda \geq 0} \max_{\pi} \quad J(\pi) - \lambda(J_C(\pi) - d)$$

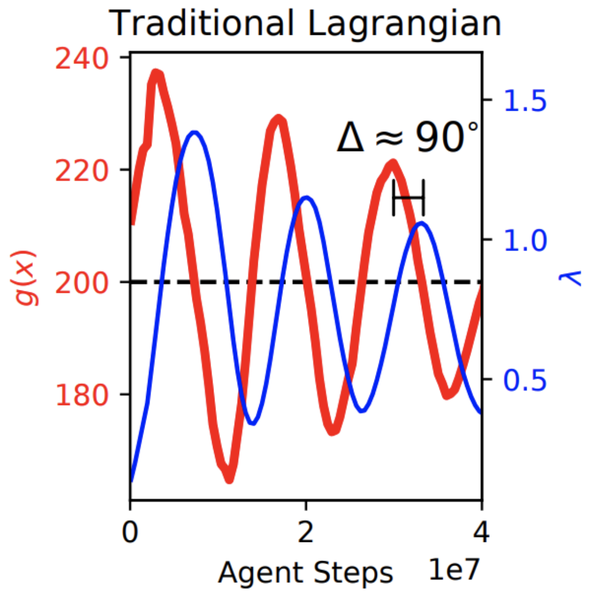
    - We need to select a proper learning rate for λ

$$\lambda_{i+1} = \max\left(0, \lambda_i + \eta(J_C(\pi_{i+1}) - d)\right)$$

Primal-dual method 2.0

- What's the problem?
  - Improper learning rate can lead to oscillation training behavior and phase shift between λ and $J_C(\pi)$

$J_C(\pi)$



Primal-dual method 2.0

- How to solve the unstable training problem?
- Denote $e_i = J_C(\pi_{i+1}) - d$, we could observe that

$$\lambda_{i+1} = \lambda_i + \eta e_i = \lambda_{i-1} + \eta(e_i + e_{i-1}) = \lambda_1 + \eta \sum_{j=1}^{i} e_j$$

- The dual problem updating is an integral controller for λ!
- Then the oscillation, phase shift problems could be explained.
- We can regard the dual variable updating as a control system and design a PID controller to solve λ.

Primal-dual method 2.0

- PID-Lagrangian algorithm

Algorithm 2 PID-Controlled Lagrange Multiplier

Choose tuning parameters: $K_P, K_I, K_D \geq 0$
Integral: $I \leftarrow 0$
Previous Cost: $J_{C,prev} \leftarrow 0$
repeat at each iteration k
    Receive cost J_C
    $\Delta \leftarrow J_C - d$
    $\partial \leftarrow (J_C - J_{C,prev})_+$
    $I \leftarrow (I + \Delta)_+$
    $\lambda \leftarrow (K_P\Delta + K_I I + K_D\partial)_+$
    $J_{C,prev} \leftarrow J_C$
    return λ

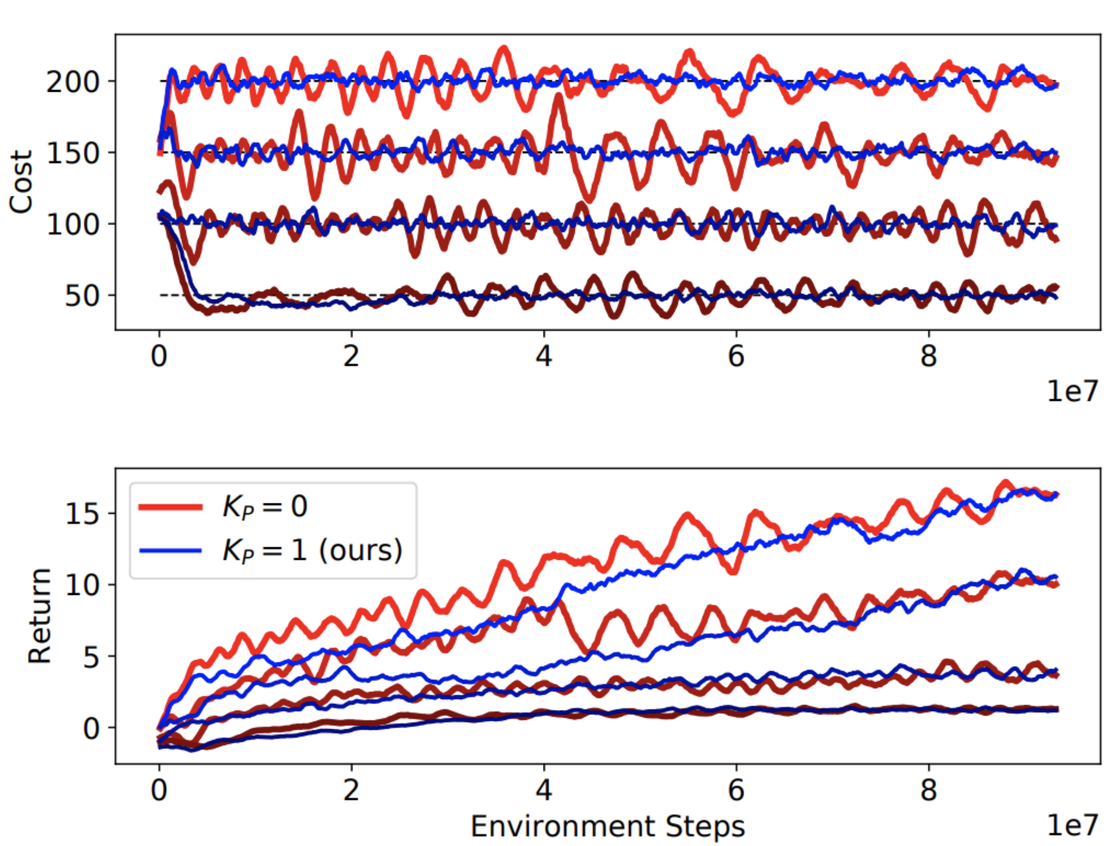
Algorithm 1
Constraint-Controlled Reinforcement Learning

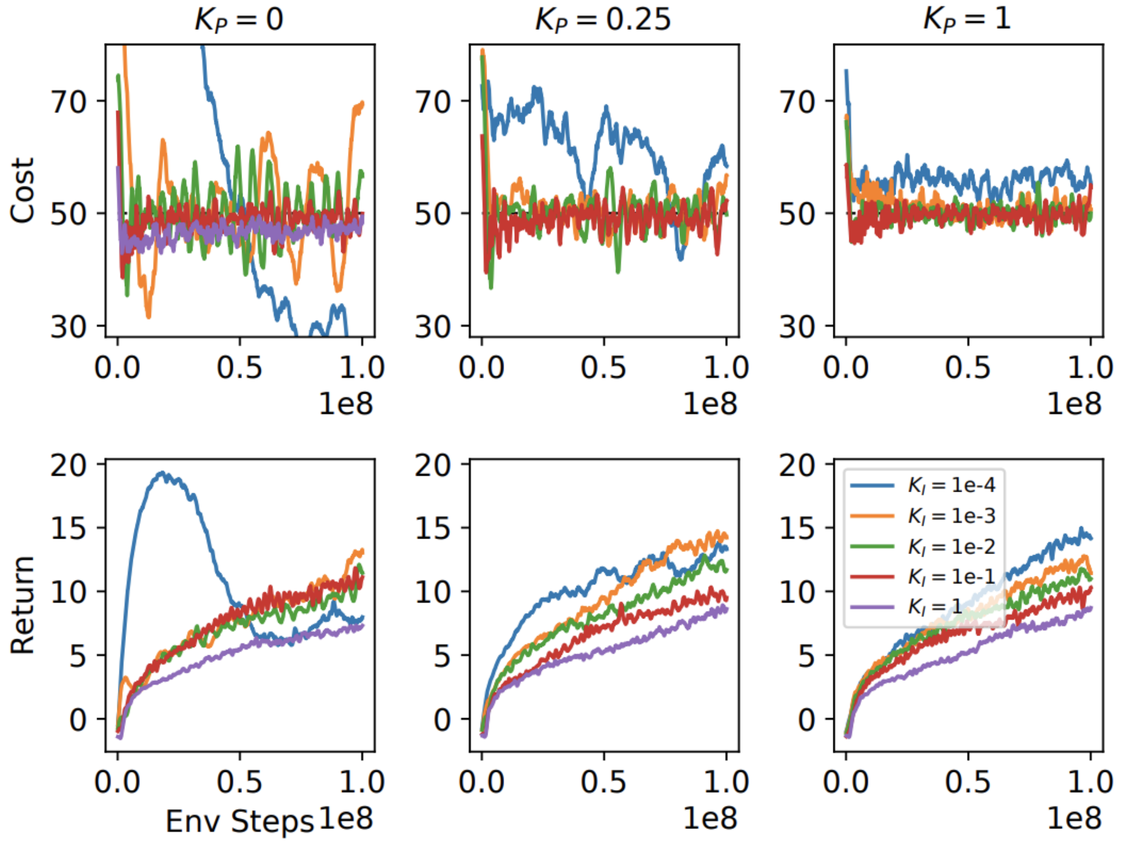
procedure CONSTRAINED RL($\pi_{\theta_0}(\cdot|s), d$)
    Initialize control rule (as needed)
    $\mathcal{J}_C \leftarrow \{\}$ ▷ cost measurement history
    repeat
        Sample environment: ▷ a minibatch
            $a \sim \pi(\cdot|s;\theta), s' \sim T(s,a),$
            $r \sim R(s,a,s'), c \sim C(s,a,s')$
        Apply feedback control:
            Store sample estimate $\hat{J}_C$ into $\mathcal{J}_C$
            $\lambda \leftarrow h(\mathcal{J}_C, d),\ \lambda \geq 0$
        Update π by RL: ▷ by Lagrangian objective
            Update critics, $V_\phi(s), V_{C,\psi}(s)$ ▷ if using
            $\nabla_\theta \mathcal{L} = \frac{1}{1+\lambda}\left(\nabla_\theta \hat{J}(\pi_\theta) - \lambda \nabla_\theta \hat{J}_C(\pi_\theta)\right)$
    until converged
    return π_θ
end procedure

Stooke, Adam, Joshua Achiam, and Pieter Abbeel. "Responsive safety in reinforcement learning by pid lagrangian methods." *International Conference on Machine Learning*. PMLR, 2020.

Primal-dual method 2.0

- PID-Lagrangian algorithm has better training stability

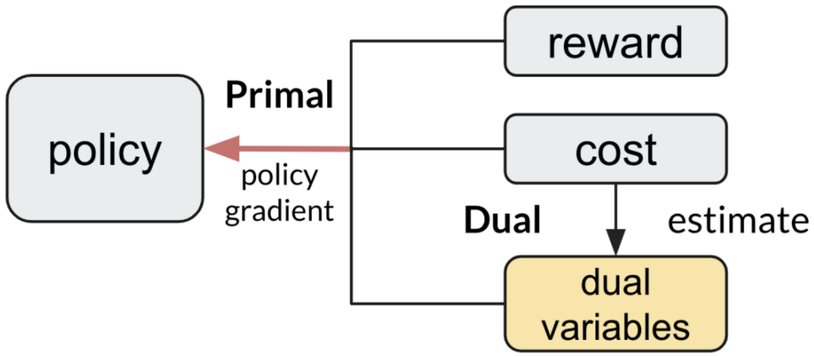




Stooke, Adam, Joshua Achiam, and Pieter Abbeel. "Responsive safety in reinforcement learning by pid lagrangian methods." *International Conference on Machine Learning*. PMLR, 2020.

Primal-dual method 3.0

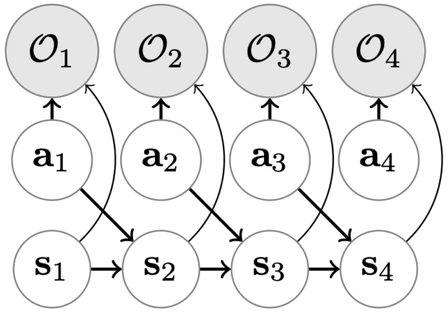
- The problems for primal-dual methods:
  - Lack of optimality guarantees both during training and after training
  - The policy only converges to a saddle point asymptotically
  - The primal problem is usually difficult to optimize



- How to overcome the theoretical drawbacks of primal-dual methods?

Variational inference approach

- Consider the safe RL problem from the probabilistic inference perspective
  - Denote $\mathcal{O}$ as the optimality variable
  - $p(\mathcal{O}_t = 1 \mid s_t, a_t) = K_{\text{norm}} \exp\big(r(s_t, a_t)\big) \propto \exp\big(r(s_t, a_t)\big)$



graphical model with optimality variables

- We have the following Evidence Lower Bound (ELBO) for likelihood:

$$\log p_\pi(\mathcal{O} = 1) = \log \int p(\mathcal{O} = 1 \mid \tau) p_\pi(\tau) d\tau \geq \mathbb{E}_{\tau \sim q}[\sum_{t=0}^{\infty} \gamma^t r_t] - \alpha KL(q \| p_\pi) = J(q, \pi)$$

Levine, Sergey. "Reinforcement learning and control as probabilistic inference: Tutorial and review." *arXiv preprint arXiv:1805.00909* (2018).

ELBO Derivation

- Denote the trajectory probability of a policy as:

$$p_{\pi_\theta}(\tau) = p(s_0)\prod_{t\geq 0} p(s_{t+1}\,|\,s_t, a_t)\pi_\theta(a_t\,|\,s_t)p(\theta) \qquad q(\tau) = p(s_0)\prod_{t\geq 0} p(s_{t+1}\,|\,s_t, a_t)q(a_t\,|\,s_t)$$

- q is the variational distribution. We have:

$$\log p_\pi(O=1) = \log \int p(O=1\mid\tau)p_\pi(\tau)d\tau$$

$$= \log \mathbb{E}_{\tau\sim q}[\frac{p(O=1\mid\tau)p_\pi(\tau)}{q(\tau)}]$$

$$\geq \mathbb{E}_{\tau\sim q}\log\frac{p(O=1\mid\tau)p_\pi(\tau)}{q(\tau)}$$ ➡ Jensen's inequality

$$= \mathbb{E}_{\tau\sim q}\log p(O=1\mid\tau) + \mathbb{E}_{\tau\sim q}\log\frac{p_\pi(\tau)}{q(\tau)}$$

$$\propto \mathbb{E}_{\tau\sim q}[\sum_{t=0}^{\infty}\gamma^t r_t] - \alpha D_{\mathrm{KL}}(q(\tau)\|p_\pi(\tau)) = \mathcal{J}(q,\pi) = \mathbb{E}_{\tau\sim q}\left[\sum_{t=0}^{\infty}\left(\gamma^t r_t - \alpha D_{\mathrm{KL}}(q(\cdot|s_t)\|\pi_\theta(\cdot|s_t))\right)\right] + \log p(\theta), \quad \forall q(a|s_t)\in\Pi_{\mathcal{Q}}^{\epsilon_1}.$$

trajectory-wise ELBO

state-wise ELBO

Levine, Sergey. "Reinforcement learning and control as probabilistic inference: Tutorial and review." *arXiv preprint arXiv:1805.00909* (2018).

Solving Variational inference (details in the paper)

- We could improve the ELBO by Expectation-Maximization (EM) algorithm

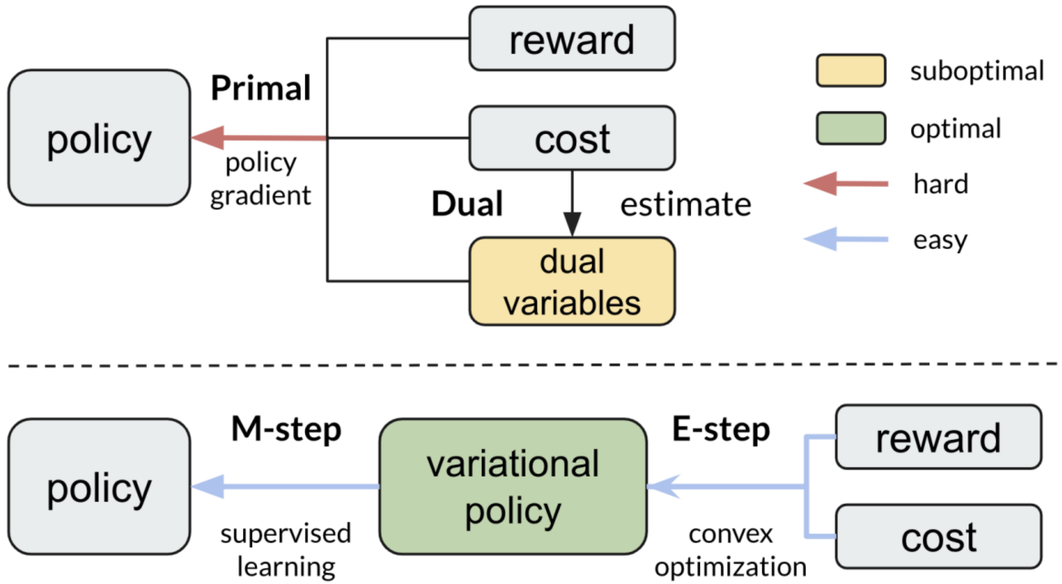
$$J(q,\pi) = \mathbb{E}_{\tau\sim q}[\sum_{t=0}^{\infty}\gamma^t r_t] - \alpha KL(q\|p_\pi)$$

- E-step: improve the variational distribution q
    - q can be solved analytically with optimality guarantee
- M-step: improve the policy π by minimizing the divergence $KL(q\|p_\pi)$
    - By restricting the updated policy within a trust-region, we could achieve robustness guarantee and worst-case performance bound

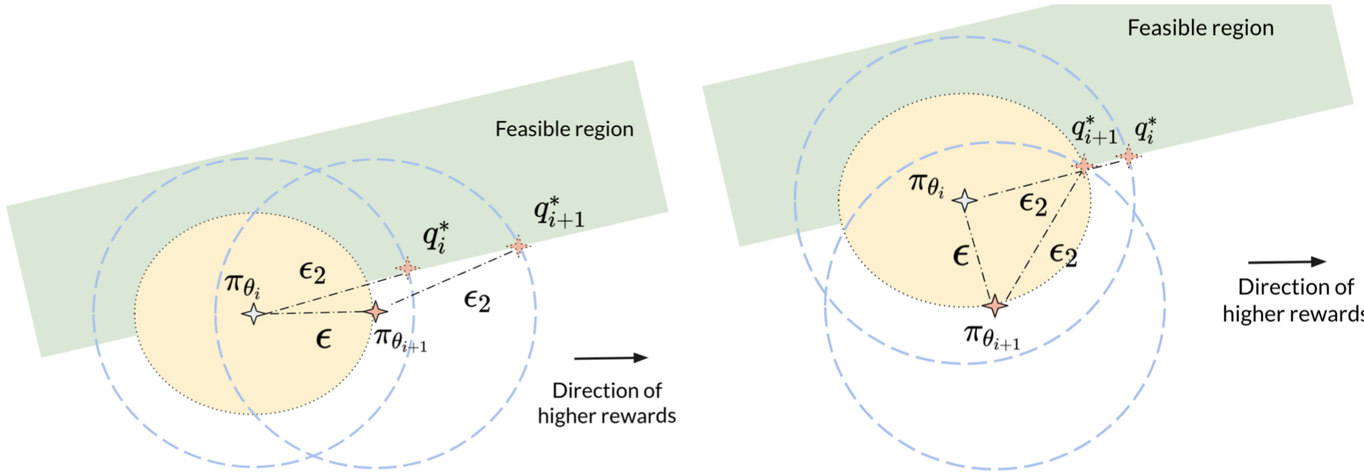
Liu, Z., Cen, Z., Isenbaev, V., Liu, W., Wu, Z.S., Li, B., & Zhao, D. (2022). Constrained Variational Policy Optimization for Safe Reinforcement Learning. *ArXiv, abs/2201.11927*.

Advantages

- Comparison to primal-dual method



- Robustness analysis



(a) Regular policy updating (b) Worst-case policy updating

Figure 7. Illustration of the policy updating at the i-th iteration under the M-step approximation error.

- Additional benefits include: sample-efficiency (off-policy), stable training performance, optimality guarantee for **each** policy update, etc…

Liu, Z., Cen, Z., Isenbaev, V., Liu, W., Wu, Z.S., Li, B., & Zhao, D. (2022). Constrained Variational Policy Optimization for Safe Reinforcement Learning. *ArXiv, abs/2201.11927*.

Worthy Reading

- Ray, Alex, Joshua Achiam, and Dario Amodei. "Benchmarking safe exploration in deep reinforcement learning." arXiv preprint arXiv:1910.01708 7 (2019).
- https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html
- Garcıa, Javier, and Fernando Fernández. "A comprehensive survey on safe reinforcement learning." Journal of Machine Learning Research 16.1 (2015): 1437-1480.