A Study of Policy Optimization in RL

Mathematical Principles of Reinforcement Learning
University of Wisconsin-Madison

Dheeraj Dhillon

Policy Optimization

- **Direct policy optimization** is a fundamentally different approach to learning policies where we parameterize a policy and learn it directly.
- It is different from value based methods which operate by estimating value functions and extracting policies indirectly.

- **What is the objective parameterized policies optimize upon?**
- The expected return, also called $J(\pi)$ and the performance of a policy π for a starting distribution rho can be written as:

$$V^{\pi}(\rho) := \mathbb{E}_{s_0 \sim \rho}[V^{\pi}(s_0)]$$

- Using this objective, the optimization for a parameterized policy is written as:

$$\max_{\theta \in \Theta} V^{\pi_\theta}(\rho)$$

Why optimize policies directly?

- **Continuous/high-dimensional actions**:
  - Value based methods require max operator which is intractable for continuous or large action spaces. Policy parameterizations avoid this by directly outputting actions.[6]

- **Policy parameterization naturally represent stochastic policies**:
  - helpful where optimal behaviour is stochastic/randomized in partially observable environments or games.

- **Function approximation compatibility**
  - Seamless integration with neural network function approximation.

- **Smooth optimization landscape**
  - This enables gradient-based optimization with better understanding of convergence properties.

The policy gradient theorem allows us to formalize this optimization.

Policy Gradient theorem

- We get several useful and structured forms for the policy gradient using the policy gradient theorem by Sutton, 1999[1]

$$\nabla V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim \pi_\theta} \quad R(\tau) \sum_{t=0}^{\infty} \nabla \log \pi_\theta(a_t \mid s_t) \qquad \text{(REINFORCE)}$$

$$= \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_\theta} [Q^{\pi_\theta} \nabla \log \pi_\theta(a \mid s)] \qquad \text{(Action-value)}$$

$$= \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_\theta}, a \sim \pi_\theta} [A^{\pi_\theta} \nabla \log \pi_\theta(a \mid s)] \qquad \text{(Advantage)}$$

Going beyond vanilla policy gradients

For vanilla policy gradient methods, Kakade and Langford [4] also showed examples where the convergence may be too slow or that the gradient would even point to wrong direction.

Hence we seek to go beyond vanilla policy gradient methods.

In a policy optimization algorithm, we want an update step that[9]

- uses rollouts collected from the most recent policy as efficiently as possible
- and takes steps that respect distance in policy space as opposed to distance in parameter space

To figure out the right update rule, we need to exploit relationships between performance of two policies. The performance difference lemma, lets us compare the relative performance of two policies

$$J(\tilde{\pi}) = J(\pi) + \mathbb{E}_{\tau \sim \tilde{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t) \right]$$

$$= J(\pi) + \mathbb{E}_{s_0, a_0, s_1, \cdots \sim \tilde{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t) \right]$$

Proof of performance difference lemma

$$
\begin{aligned}
J(\tilde{\pi}) - J(\pi) &= \mathbb{E}_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t A^{\pi}(s_t,a_t)\right] \\
&= \mathbb{E}_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t\Big(Q^{\pi}(s_t,a_t) - V^{\pi}(s_t)\Big)\right] \\
&= \mathbb{E}_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t\Big(R(s_t,a_t) + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)\Big)\right] \\
&= \mathbb{E}_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\right] + \mathbb{E}_{\tau\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty}\Big(\gamma^{t+1}V^{\pi}(s_{t+1}) - \gamma^t V^{\pi}(s_t)\Big)\right] \\
&= J(\tilde{\pi}) + \mathbb{E}_{\tau\sim\tilde{\pi}}\left[\sum_{t=1}^{\infty}\gamma^t V^{\pi}(s_t) - \sum_{t=0}^{\infty}\gamma^t V^{\pi}(s_t)\right] \\
&= J(\tilde{\pi}) - \mathbb{E}_{\tau\sim\tilde{\pi}}[V^{\pi}(s_0)] \\
&= J(\tilde{\pi}) - J(\pi).
\end{aligned}
$$

Looking closer at the performance difference lemma

$$J(\tilde{\pi}) = J(\pi) + \mathbb{E}_{s_0,a_0,s_1,\cdots\sim\tilde{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t A^{\pi}(s_t,a_t)\right]$$

$$= J(\pi) + \sum_{t=0}^{\infty}\sum_{s} P(s_t = s \mid \tilde{\pi}) \sum_{a}\tilde{\pi}(a \mid s)\gamma^t A_{\pi}(s,a)$$

$$= J(\pi) + \sum_{s}\sum_{t=0}^{\infty}\gamma^t P(s_t = s \mid \tilde{\pi}) \sum_{a}\tilde{\pi}(a \mid s) A_{\pi}(s,a)$$

$$J(\tilde{\pi}) = J(\pi) + \sum_{s}\sum_{t=0}^{\infty}\gamma^t P(s_t = s \mid \tilde{\pi}) \sum_{a}\tilde{\pi}(a \mid s) A_{\pi}(s,a)$$

$$= J(\pi) + \frac{1}{1-\gamma}\sum_{s} d_{\tilde{\pi}}(s) \sum_{a}\tilde{\pi}(a \mid s) A_{\pi}(s,a).$$

Analyzing the policy performance difference can help us gain insights into finding a more helpful optimization objective

We can rewrite the relative policy performance identity using have the discounted state-visitation distribution defined as shown.

$$d_{\pi}(s) = (1-\gamma)\big(P(s_0 = s \mid \pi) + \gamma P(s_1 = s \mid \pi) + \gamma^2 P(s_2 = s \mid \pi) + \ldots$$
$$= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_t = s \mid \pi)$$

A nice feature of this optimization problem: defines the performance of pi' in terms of the advantages from pi!

Almost there! But, problematic feature: still requires trajectories sampled from pi'....

Connecting the lemma with policy iteration

$$J(\tilde{\pi}) = J(\pi) + \frac{1}{1-\gamma} \sum_s d_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a \mid s) A_\pi(s,a).$$

- This equation implies that any policy update from pi to pi hat has a nonnegative expected advantage at every state s, is guaranteed to increase the policy performance, or leave it constant if advantage is zero everywhere.
- This implies the classic result that the update performed by exact policy iteration, which uses the deterministic policy iteration, improves the policy if there is at least one state-action pair with a positive advantage value and nonzero state visitation probability, otherwise algorithm converged to the optimal policy. as shown in immediate next equation:

$$\sum_a \pi_{\text{greedy}}(a \mid s) A_\pi(s,a) = \max_a A_\pi(s,a)$$

$$\begin{aligned}
\pi_{\text{greedy}}(s) &:= \text{argmax}_a Q_\pi(s,a) && \text{(Policy iteration step)} \\
&= \text{argmax}_a \left(Q_\pi(s,a) - V_\pi(s)\right) \\
&= \text{argmax}_a A_\pi(s,a) && \text{(Using advantage definition)}
\end{aligned}$$

$$\begin{aligned}
J(\pi_{\text{greedy}}) - J(\pi) &= \frac{1}{1-\gamma} \sum_s d_{\pi_{\text{greedy}}}(s) \sum_a \pi_{\text{greedy}}(a \mid s) A_\pi(s,a) \\
&= \frac{1}{1-\gamma} \sum_s d_{\pi_{\text{greedy}}}(s) \max_a A_\pi(s,a) \\
&\geq 0.
\end{aligned}$$

- However, in the approximate setting, it will typically be unavoidable, due to estimation and approximation error, that there will be some states s for which the expected advantage is negative.
- The complex dependency of dpi hat s on pi hat makes above equation difficult to optimize directly. we make an approximation to it then, this approximation is also called the “surrogate”...

Performance difference and the approximation to performance difference

- What if we just said d^pi' ~= d^pi and didn't worry about it?
- Turns out this approximation is pretty good when pi' and pi are close! But why, and how close do they have to be?

$$J(\tilde{\pi}) = J(\pi) + \frac{1}{1-\gamma} \sum_s d_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a \mid s) A_{\pi}(s,a).$$

$$L_{\pi}(\tilde{\pi}) = J(\pi) + \frac{1}{1-\gamma} \sum_s d_{\pi}(s) \sum_a \tilde{\pi}(a \mid s) A_{\pi}(s,a).$$

$$\tilde{\pi} = \pi_\theta, \quad \pi = \pi_0$$

$$J(\pi_\theta) - L_{\pi_{\theta_0}}(\pi_\theta) = \frac{1}{1-\gamma} \sum_s (d_{\pi_\theta}(s) - d_{\pi_{\theta_0}}(s)) \sum_a \pi_\theta(a \mid s) A_{\pi_{\theta_0}}(s,a).$$

Can we say something about how close is the actual performance to the approximation?

$$\nabla_\theta \left(J(\pi_\theta) - L_{\pi_{\theta_0}}(\pi_\theta) \right) = \nabla_\theta \left[\frac{1}{1-\gamma} \sum_s (d_{\pi_\theta}(s) - d_{\pi_{\theta_0}}(s)) \sum_a \pi_\theta(a \mid s) A_{\pi_{\theta_0}}(s,a) \right]$$

$$= \frac{1}{1-\gamma} \sum_s \left[(\nabla_\theta d_{\pi_\theta}) \left(\sum_a \pi_\theta(a \mid s) A_{\pi_{\theta_0}}(s,a) \right) + \left(d_{\pi_\theta}(s) - d_{\pi_{\theta_0}}(s) \right) \left(\nabla_\theta \sum_a \pi_\theta(a \mid s) A_{\pi_{\theta_0}}(s,a) \right) \right]$$

Since, $\sum_a \pi_{\theta_0}(a \mid s) A_{\pi_{\theta_0}}(s,a) = 0$,

we get at $\theta = \theta_0$:

$$L_{\pi_{\theta_0}}(\pi_{\theta_0}) = J(\pi_{\theta_0}),$$

$$\nabla_\theta L_{\pi_{\theta_0}}(\pi_\theta)\big|_{\theta=\theta_0} = \nabla_\theta J(\pi_\theta)\big|_{\theta=\theta_0}.$$

Conservative Policy Iteration

- CPI performs each iteration in a mixture update of the form:[4]

$$\pi_{\text{new}}(a \mid s) = (1-\alpha)\pi_{\text{old}}(a \mid s) + \alpha\pi'(a \mid s).$$

- Here pi' is approximately greedy policy and can be calculated by solving regression problem.
- This gives us the bound on performance:

$$J(\pi_{\text{new}}) \geq L_{\pi_{\text{old}}}(\pi_{\text{new}}) - \frac{2\epsilon\gamma}{(1-\gamma)^2}\alpha^2$$

$$\text{where} \epsilon = \max_s \left| \mathbb{E}_{a\sim\pi'(a|s)} \left[A_\pi(s,a) \right] \right|$$

- Above bound, which applied to conservative policy iteration, implies that a policy update that improves the right-hand side is guaranteed to improve the true performance J and has structured and promising convergence properties.

- CPI also has the advantage that we don't need to take gradients at any step, as long as we can calculate the approximately greedy policy.

Going from constrained mixture updates to constrained generalized updates

- In CPI, we took a linear combination to limit big changes.
- To generalize this more, because mixture policies are not used much, we can take inspiration from the bounds of the CPI itself, the policy parameters differ havin upper bound of alpha.
- have the total variation divergence, which is defined as:

$$D_{\mathrm{TV}}(p||q) = \frac{1}{2}\sum_i |p_i - q_i|$$

- For CPI, we can observe the total variation divergence between new and old policies as:

$$\begin{aligned}
2D_{\mathrm{TV}}\left(\pi_{\mathrm{new}}(\cdot \mid s)||\pi_{\mathrm{old}}(\cdot \mid s)\right) &= \sum_a \left|\pi_{\mathrm{new}}(\cdot \mid s) - \pi_{\mathrm{old}}(\cdot \mid s)\right| \\
&= \|(1-\alpha)\pi_{\mathrm{old}}(\cdot \mid s) + \alpha\pi'(\cdot \mid s) - \pi_{\mathrm{old}}(\cdot \mid s)\|_1 \\
&= \| - \alpha\pi_{\mathrm{old}}(\cdot \mid s) + \alpha\pi'(\cdot \mid s)\|_1 \\
&\leq \alpha\|\pi_{\mathrm{old}}(\cdot \mid s)\|_1 + \alpha\|\pi'(\cdot \mid s)\|_1 \qquad \text{(using triangle inequality)} \\
&= 2\alpha
\end{aligned}$$

- What if we just had the variation divergence as the constraint for policy updates instead of mixture policies? This is the idea Schulman et al[2] propose with TRPO.

TRPO generalizes the constraint using total variation divergence and provides policy improvement bounds for it

- Let: $\alpha = D_{\mathrm{TV}}^{\max}(\pi_{\mathrm{old}}, \pi_{\mathrm{new}})$ where $D_{\mathrm{TV}}^{\max}(\pi_{\mathrm{old}}, \pi_{\mathrm{new}}) = \max_s D_{\mathrm{TV}}(\pi_{\mathrm{old}}(\cdot \mid s) || \pi_{\mathrm{new}}(\cdot \mid s))$

- Then a form of constraint we can keep would be having the old and new policies to be alpha coupled which basically means that the sampling from two policies agrees with probability at least 1-alpha.
- With this assumption, we are able to derive bounds for policy improvement in this general case as we did in the mixture policies case:

$$J(\pi_{\mathrm{new}}) \geq L_{\pi_{\mathrm{old}}}(\pi_{\mathrm{new}}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}\alpha^2, \qquad \text{where} \quad \epsilon = \max_{s,a} |A_\pi(s,a)|$$

- This is the principal theoretical result provided by the TRPO paper and proved in detail there.
- The surrogate L here is again the same as we used in CPI case.

A KL divergence based bound

- Pinsker's inequality between the total variation divergence and Kl divergence which allows us to write the the theoretical performance improvement in terms of Kl divergence from the total variation divergence bound.

$$D_{\mathrm{TV}}(p||q)^2 \leq D_{\mathrm{KL}}(p||q)$$

- The updated policy improvement bound we get:

$$J(\pi_{\mathrm{new}}) \geq L_{\pi_{\mathrm{old}}}(\pi_{\mathrm{new}}) - C\left(D_{\mathrm{TV}}^{\max}(\pi_{\mathrm{old}}, \pi_{\mathrm{new}})\right)^2 \qquad \text{where } C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$$

$$\geq L_{\pi_{\mathrm{old}}}(\pi_{\mathrm{new}}) - CD_{\mathrm{KL}}^{\max}(\pi_{\mathrm{old}}, \pi_{\mathrm{new}}).$$

How to make use of the bound?

At iteration step i we have:

$$J(\pi) \geq L_{\pi_i}(\pi) - CD_{\mathrm{KL}}^{\max}(\pi_i, \pi)$$

let $M_i(\pi) := L_{\pi_i}(\pi) - CD_{KL}^{\max}(\pi_i, \pi)$

then, for every π:

$$J(\pi) \geq M_i(\pi)$$

and for $\pi = \pi_i$:

$$M_i(\pi_i) = L_{\pi_i}(\pi_i) - C \cdot 0 = J(\pi_i). \qquad \text{(using our definition for } L_{\pi_i}(\pi))$$

- How to make use of the bound to obtain next step policy so that we have monotonically improving policies? **Take argmax!**

$$\pi_{i+1} = \mathrm{argmax}_{\pi} M_i(\pi)$$

- Then

$$J(\pi_{i+1}) \geq M_i(\pi_{i+1}) \geq M_i(\pi_i) = J(\pi_i).$$

- we get the next policy which improves in performance or stays the same.

A policy iteration algorithm

Algorithm 1 Policy iteration algorithm with guaranteed non-decreasing expected return J

Initialize π_0.

for $i = 0, 1, 2, \ldots$ until convergence **do**

compute all advantage values $A_{\pi_i}(s, a)$.

Solve the constrained optimization problem

$$\pi_{i+1} = \arg\max_{\pi} \left[L_{\pi_i}(\pi) - C D_{\mathrm{KL}}^{\max}(\pi_i, \pi) \right]$$

where $C = \dfrac{4\epsilon\gamma}{(1-\gamma)^2}$

and $L_{\pi_i}(\pi) = \eta(\pi_i) + \dfrac{1}{1-\gamma} \sum_s d_{\pi_i}(s) \sum_a \pi(a \mid s) A_{\pi_i}(s, a)$

end for

- Applying approximations to the above theoretically justified procedure yields a practical algorithm.
- That practical algorithm retains the policy performance improvement guarantees and yields strong empiricals results and is known as **Trust Region Policy Optimization**.

Why do we need approximations to the theoretical result?

$$\max_{\pi} L_{\pi_i}(\pi) - C D_{\mathrm{KL}}^{\max}(\pi_i, \pi) \qquad \text{where } C = 4\epsilon\gamma/(1-\gamma)^2$$

- A problem is that the penalty coefficient C recommend by the theory is quite high when gamma is near 1 which makes the learning steps very small.
- A solution is to instead of using a KL penalty, use a KL constraint (called trust region)
- In this way we can take larger steps in a robust way and also control worst-case error through the constrained upper limit.

$$\max_{\pi} \quad L_{\pi_i}(\pi) \quad \text{subject to} \quad \max_{s} D_{\mathrm{KL}}(\pi_i(\cdot \mid s), \pi(\cdot \mid s)) \leq \delta$$

- Further, the max constraint is hard to compute so to make it tractable we approximate using the mean KL divergence:

$$\max_{\pi} \quad L_{\pi_i}(\pi) \quad \text{subject to} \quad \mathbb{E}_{s \sim d_{\pi_i}} D_{\mathrm{KL}}(\pi_i(\cdot \mid s) || \pi(\cdot \mid s)) \leq \delta$$

Solving the nonlinear optimization

So we have this optimization problem after making the approximations:

$$\pi_{i+1} = \text{argmax}_{\pi} \quad L_{\pi_i}(\pi)$$
$$\text{s.t} \quad \mathbb{E}_{s\sim d_{\pi_i}} D_{\text{KL}}(\pi_i(\cdot \mid s)||\pi(\cdot \mid s)) \leq \delta$$

This policy optimization step satisfies the two constraints we wanted:

1. The objective and constraint can be estimated using rollouts from the most recent policy - efficient!

2. From the constraint, steps respect (a notion of) distance in policy space! The update is parameterization invariant.

Parameterizing our policies we can write as:

$$\theta_{\text{new}} = \text{argmax}_{\theta} \quad L_{\theta_{\text{old}}}(\theta) \quad \text{s.t.} \quad \mathbb{E}_{s\sim d_{\theta_{\text{old}}}} [D_{\text{KL}}(\theta_{\text{old}}(\cdot \mid s)||\theta(\cdot \mid s))] \leq \delta$$

This is a highly nonlinear optimization problem and directly solving it is highly intractable.

So how do we solve it? **Approximately!**

Analyzing the objective

$$\max_\theta \quad \sum_s d_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a \mid s) Q_{\theta_{\text{old}}}(s,a) \quad \text{s.t.} \quad \mathbb{E}_{s \sim d_{\theta_{\text{old}}}} \left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot \mid s) || \pi_\theta(\cdot \mid s)) \right] \leq \delta$$

- We can see that the V term does not affect the optimization, although using V function or a baseline helps reduce variance.

$$\sum_s d_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a \mid s) A_{\theta_{\text{old}}}(s,a) = \sum_s d_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a \mid s) \left[Q_{\theta_{\text{old}}}(s,a) - V_{\theta_{\text{old}}}(s) \right]$$

$$\sum_s \rho_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a \mid s) V_{\theta_{\text{old}}}(s) = \sum_s \rho_{\theta_{\text{old}}}(s) V_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a \mid s)$$

$$= \sum_s \rho_{\theta_{\text{old}}}(s) V_{\theta_{\text{old}}}(s) \cdot 1$$

- Using an importance sampling estimator, we can make use of samples we already have using our old distribution.

$$\max_\theta \quad \mathbb{E}_{s \sim d_{\theta_{\text{old}}}, a \sim q} \left[\frac{\pi_\theta(a \mid s)}{q(a \mid s)} Q_{\theta_{\text{old}}}(s,a) \right] \quad \text{s.t.} \quad \mathbb{E}_{s \sim d_{\theta_{\text{old}}}} \left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot \mid s) || \pi_\theta(\cdot \mid s)) \right] \leq \delta$$

Linear approximation to the surrogate objective and quadratic approximation to the mean KL constraint

- **Objective**: computing linear approximation using taylor series expansion.

For the current policy parameters θ_{old}, we take a small step to arrive at the new policy parameters which we write as: $\theta := \theta_{\text{old}} + \Delta\theta$. The linear approximation of the objective $L_{\pi_{\text{old}}}$ becomes:

$$\begin{aligned} L_{\theta_{\text{old}}}(\theta) &= L_{\theta_{\text{old}}}(\theta_{\text{old}} + \Delta\theta) \\ &\approx L_{\theta_{\text{old}}}(\theta_{\text{old}}) + g^\top \Delta\theta, \qquad \text{where } g := \nabla_\theta L_{\theta_{\text{old}}}(\theta)\Big|_{\theta=\theta_{\text{old}}}. \end{aligned}$$

- **Constraint**: taking quadratic approximation

The mean KL constraint we approximate quadratically because the first-order term for the mean KL constraint becomes zero at θ_{old}.

$$\begin{aligned} \mathbb{E}_{s\sim d_{\theta_{\text{old}}}}\left[D_{\text{KL}}(\theta_{\text{old}}(\cdot \mid s)||\theta(\cdot \mid s))\right] &=: \bar{D}_{\text{KL}}(\theta_{\text{old}}, \theta) \\ &= \bar{D}_{\text{KL}}(\theta_{\text{old}}, \theta_{\text{old}} + \Delta\theta) \\ &\approx \frac{1}{2}\Delta\theta^\top A \Delta\theta \end{aligned}$$

Here A is the Hessian of the mean KL divergence measure.

The optimization takes form of that of Natural Policy Gradient algorithm

- Our optimization takes the following form, which also arises exactly in NPG:

$$\theta_{\text{new}} = \text{argmax}_\theta \quad g^\top \Delta\theta \qquad \text{s.t.} \quad \frac{1}{2}\Delta\theta^\top A \Delta\theta \leq \delta$$

Let $d := \Delta\theta = \theta - \theta_{\text{old}}$. Then for $\lambda > 0$, the lagrangian becomes:

$$\mathcal{L}(d, \lambda) = g^\top d - \lambda\left(\frac{1}{2} d^\top A d - \delta\right)$$

Taking gradient w.r.t. d and setting to zero:

$$\nabla_d \mathcal{L}(d, \lambda) = g - \lambda A d = 0 \implies \lambda A d = g.$$

Assuming A is invertible gives the direction:

$$d = \Delta\theta = \frac{1}{\lambda} A^{-1} g.$$

The point to note is that the optimizer points in the direction as given:

$$\theta - \theta_{\text{old}} \propto A^{-1} g. \qquad \text{where } A$$

- We can read this that the optimization is choosing the steepest direction under the geometry induced by KL divergence. The search direction for this optimization can be computed using lagrange multipliers.

- The NPG direction A^-1 g is covariant, i.e., it points in the same direction regardless of the parameterization used to compute it.

Exact solution lies on the ellipsoid boundary

$$\frac{1}{2}d^\top A d = \delta.$$

Substituting $d = \frac{1}{\lambda}A^{-1}g$ gives:

$$\begin{aligned}
\delta &= \frac{1}{2}\left(\frac{1}{\lambda}A^{-1}g\right)^\top A\left(\frac{1}{\lambda}A^{-1}g\right) \\
&= \frac{1}{2\lambda^2}(A^{-1}g)^\top A(A^{-1}g) \\
&= \frac{1}{2\lambda^2}g^\top A^{-1}AA^{-1}g \qquad \text{(using } A\text{'s symmtericity)} \\
&= \frac{1}{2\lambda^2}g^\top A^{-1}g
\end{aligned}$$

Rearranging we can write λ as:

$$\lambda = \sqrt{\frac{2\delta}{g^\top A^{-1}g}}$$

Plugging this back in $d = \frac{1}{\lambda}A^{-1}g$:

$$d^\star = \sqrt{\frac{2\delta}{g^\top A^{-1}g}}A^{-1}g$$

- Among all directions that stay inside the trust region forming an ellipsoid, the one giving largest inner product with g is the one lying on the boundary of the ellipsoid.
- Whenever g^T Delta theta > 0. we can keep pushing outward until we hit the boundary, hence the result.
- The gradient of L is equal to the policy gradient, so this update gives a policy gradient algorithm where we pre-multiply by A^-1.

Single path sampling procedure

Algorithm 2 Monte carlo estimation of Q values through sampling trajectories of state-action pairs using TRPO's Single-Path method

Require: $\pi_{\theta_{\text{old}}}$, ρ_0, γ, horizon T, number of sampled trajectories N
Ensure: sampled state-action pairs and estimates of $Q_{\theta_{\text{old}}}$

Set $q(a \mid s) \leftarrow \pi_{\theta_{\text{old}}}(a \mid s)$
Initialize $\mathcal{D} \leftarrow \emptyset$
for $n = 1, 2, \ldots, N$ **do**
    Sample $s_0 \sim \rho_0$
    for $t = 0, 1, \ldots, T-1$ **do**
        Sample $a_t \sim q(\cdot \mid s_t)$
        Simulate one step to obtain r_t, s_{t+1}
    end for
    for $t = 0, 1, \ldots, T-1$ **do**
        $\widehat{Q}_{\theta_{\text{old}}}(s_t, a_t) \leftarrow \sum_{l=t}^{T-1} \gamma^{l-t} r_l$
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, \widehat{Q}_{\theta_{\text{old}}}(s_t, a_t))\}$
    end for
end for
return $\mathcal{D}$

- There are two methods shown for it in the TRPO paper namely single path and vine. we discuss single path in detail here.

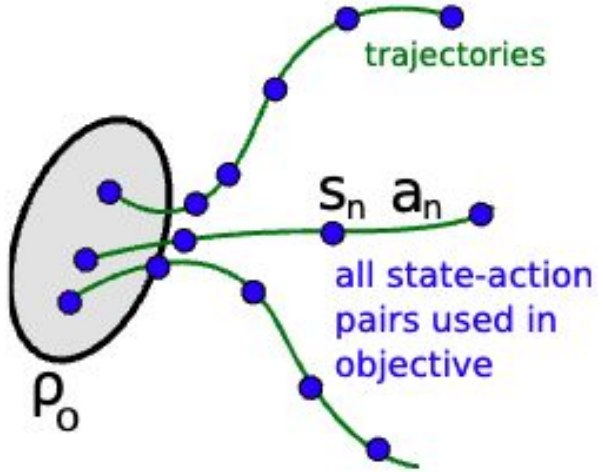


Diagram from trpo paper of one of the two sampling methods. [ref2]

Empirical estimation of objective and constraint

Algorithm 3 Constructing empirical TRPO Objective and Constraint from samples collected using Single-path procedure

Require: old policy $\pi_{\theta_{\text{old}}}$, candidate policy π_θ

Require: sample set from single-path procedure

Ensure: empirical surrogate objective $\widehat{L}(\theta)$ and empirical KL constraint $\widehat{D}_{\text{KL}}(\theta_{\text{old}}, \theta)$

1: Let $\mathcal{D} = \{(s_n, a_n, \widehat{Q}_n)\}_{n=1}^{M}$

2: Set

$$\widehat{L}(\theta) \leftarrow \frac{1}{M} \sum_{n=1}^{M} \frac{\pi_\theta(a_n \mid s_n)}{\pi_{\theta_{\text{old}}}(a_n \mid s_n)} \hat{Q}_n$$

3: Set

$$\widehat{D}_{\text{KL}}(\theta_{\text{old}}, \theta) \leftarrow \frac{1}{M} \sum_{n=1}^{M} D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot \mid s_n) \,\|\, \pi_\theta(\cdot \mid s_n))$$

- Methods such as GAE are helpful for better advantages estimates.
- In our study we have used action-value form of policy gradients.

Policy gradient estimation

$$\nabla_\theta \widehat{L}(\theta)\big|_{\theta=\theta_{\text{old}}} = \frac{1}{N}\sum_{n=1}^{N} \frac{\nabla_\theta\big(\pi_\theta(a_n \mid s_n)\ \big|_{\theta=\theta_{\text{old}}}}{\pi_{\theta_{\text{old}}}(a_n \mid s_n)} \widehat{Q}_n$$

$$= \frac{1}{N}\sum_{n=1}^{N} \frac{\pi_\theta(a_n \mid s_n)\nabla_\theta(\log(\pi_\theta(a_n \mid s_n)))}{\pi_{\theta_{\text{old}}}(a_n \mid s_n)}^{\theta=\theta_{\text{old}}} \widehat{Q}_n \quad \text{(using log-derivative identity)}$$

$$= \frac{1}{N}\sum_{n=1}^{N} \nabla_\theta(\log(\pi_\theta(a_n \mid s_n)))\Big|_{\theta=\theta_{\text{old}}} \widehat{Q}_n$$

So we get the policy gradient estimate $\widehat{g} = \nabla_\theta \widehat{L}(\theta)\big|_{\theta=\theta_{\text{old}}}$ where $\nabla_\theta \log(\pi_\theta(a_n \mid s_n))|_{\theta=\theta_{\text{old}}}$ can be computed using backprop. We can note that this takes an estimation form of the policy gradient theorem.

Estimating the Hessian of the KL divergence

The hessian of the KL divergence we can write using averaging of samples via monte carlo estimation of the KL divergence which we compute as:

$$\widehat{D}_{\mathrm{KL}}(\theta_{\mathrm{old}}, \theta) := \frac{1}{N} \sum_{n=1}^{N} D_{\mathrm{KL}}(\pi_{\theta_{\mathrm{old}}}(\cdot \mid s_n) \,\|\, \pi_{\theta}(\cdot \mid s_n))$$

Then, A, the hessian of KL divergence, is estimated as $\widehat{A}$, and takes values that we analytically compute element-wise as:

$$\widehat{A}_{ij} := \frac{1}{N} \sum_{n=1}^{N} \frac{\partial^2}{\partial \theta_i \partial \theta_j} D_{\mathrm{KL}}(\pi_{\theta_{\mathrm{old}}}(\cdot \mid s_n) \,\|\, \pi_{\theta}(\cdot \mid s_n))$$

The analytic estimator integrates over the action at each state s_n, and does not depend on the action a_n that was sampled. We are doing above method because it has computational benefits in the large scale setting, since it removes the need to store a dense Hessian or all policy gradients from a batch of trajectories.

In the other computationally hard case would have been computing covariance matrix of the gradients which would have taken the following form:

$$\widehat{A}_{ij} := \frac{1}{N} \sum_{n=1}^{N} \frac{\partial}{\partial \theta_i} \log \pi_{\theta}(a_n \mid s_n) \frac{\partial}{\partial \theta_j} \log \pi_{\theta}(a_n \mid s_n)$$

- The KL divergence Hessian A is equal to a special matrix called the Fisher Information matrix.

Natural Policy Gradient

Algorithm 2 Natural Policy Gradient

Input: initial policy parameters θ_0

for $i = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_i$ on policy $\pi_i = \pi(\theta_i)$

Estimate advantages $\hat{A}^{\pi_i}$ using any advantage estimation algorithm

Form sample estimates for

- policy gradient $\hat{g}_i$ (using advantage estimates)
- and KL-divergence Hessian (Fisher Information Matrix) $\hat{A}_i$

Compute Natural Policy Gradient update

$$\theta_{i+1} = \theta_i + \sqrt{\frac{2\delta}{\hat{g}_i^\top \hat{A}_i^{-1} \hat{g}_i}} \hat{A}_i^{-1} \hat{g}_i$$

end for

- Instead of the exact step size calculated above, NPG algorithm can also have any suitable step size chosen empirically.

What are limitations of NPG and how does TRPO overcome them?

- One problem with NPG update is that it might not be robust to trust region size delta; at some iterations delta may be too large and performance can degrade because of quadratic approximation, KL-divergence constraint may be violated
- **TRPO** uniquely implements the solution to this by enforcing KL constraint and requiring improvement in surrogate to avoid catastrophic steps. How? **Backtracking line search**.

Algorithm 3 Line search for TRPO

Compute proposed policy step $\Delta_i = \sqrt{\frac{2\delta}{\hat{g}_i^\top \hat{A}_i^{-1} \hat{g}_i}} \hat{A}_i^{-1} \hat{g}_i$

for $j = 0, 1, 2, \ldots$ L **do**

  Compute proposed update $\theta = \theta_i + \alpha^j \Delta_i$

  Solve the constrained optimization problem

  if $L_{\theta_i}(\theta) \geq 0$ and $\bar{D}_{\mathrm{KL}}(\theta_i || \theta) \leq \delta$ **then**

    accept the update and set $\theta_{i+1} = \theta_i + \alpha^j \Delta_i$

    break

  end if

end for

- Also for neural networks, number of parameters N is large- thousands or millions. Hessian has size N^2 (expensive to store) and matrix inversion complexity is O(N^3).
- The **conjugate gradient** (CG) algorithm helps to compute H^-1 g without inverting H.

Trust Region Policy Optimization

Algorithm 4 Trust Region Policy Optimization

Input: initial policy parameters θ_0

for $i = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_i$ on policy $\pi_i = \pi(\theta_i)$

Estimate advantages $\hat{A}^{\pi_i}$ using any advantage estimation algorithm

Form sample estimates for

- policy gradient $\hat{g}_i$ (using advantage estimates)
- and KL-divergence Hessian-vector product function $f(v) = \hat{A}_i v$

Use Conjugate gradient with n_{cg} iterations to obtain $x \approx \hat{A}_i^{-1} \hat{g}_i$

Estimate proposed step $\Delta_i \approx \sqrt{\frac{2\delta}{x_i^\top \hat{A}_i x_i}} x_i$

Perform backtracking line search with exponential decay to obtain final update

$$\theta_{i+1} = \theta_i + \alpha^j \Delta_i$$

end for

Experiments

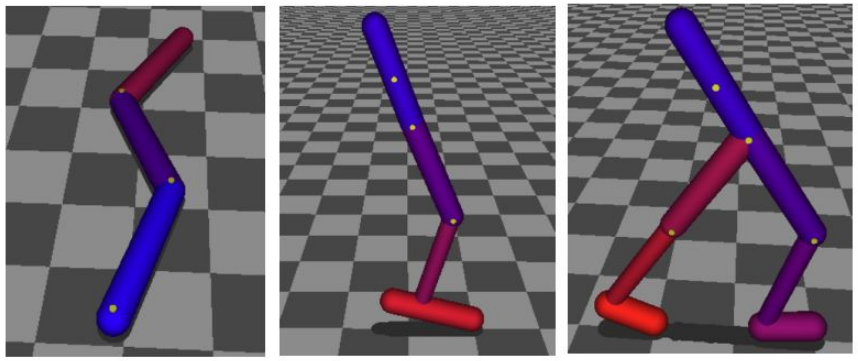
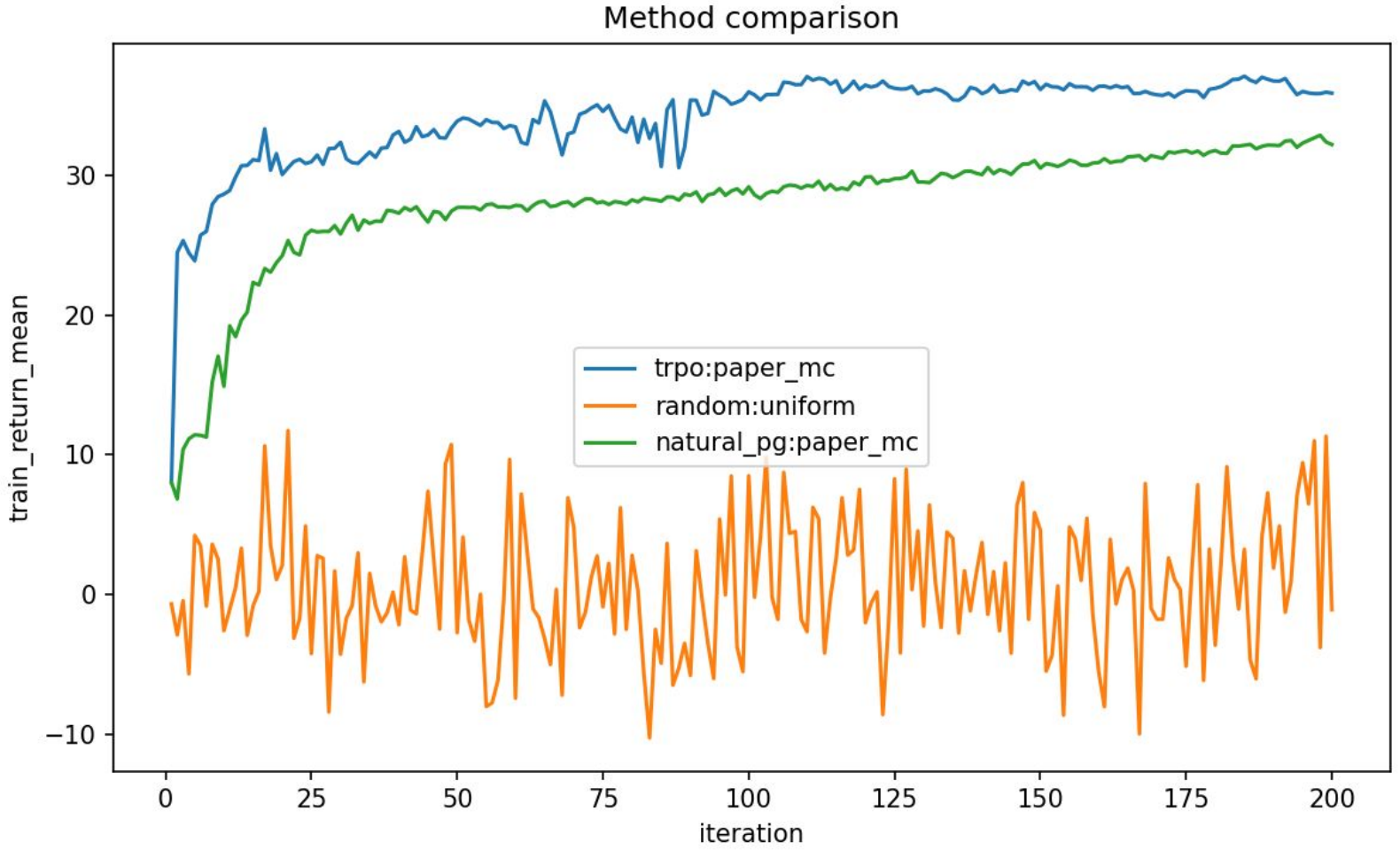


Figure 2. 2D robot models used for locomotion experiments. From left to right: swimmer, hopper, walker. The hopper and walker present a particular challenge, due to underactuation and contact discontinuities.

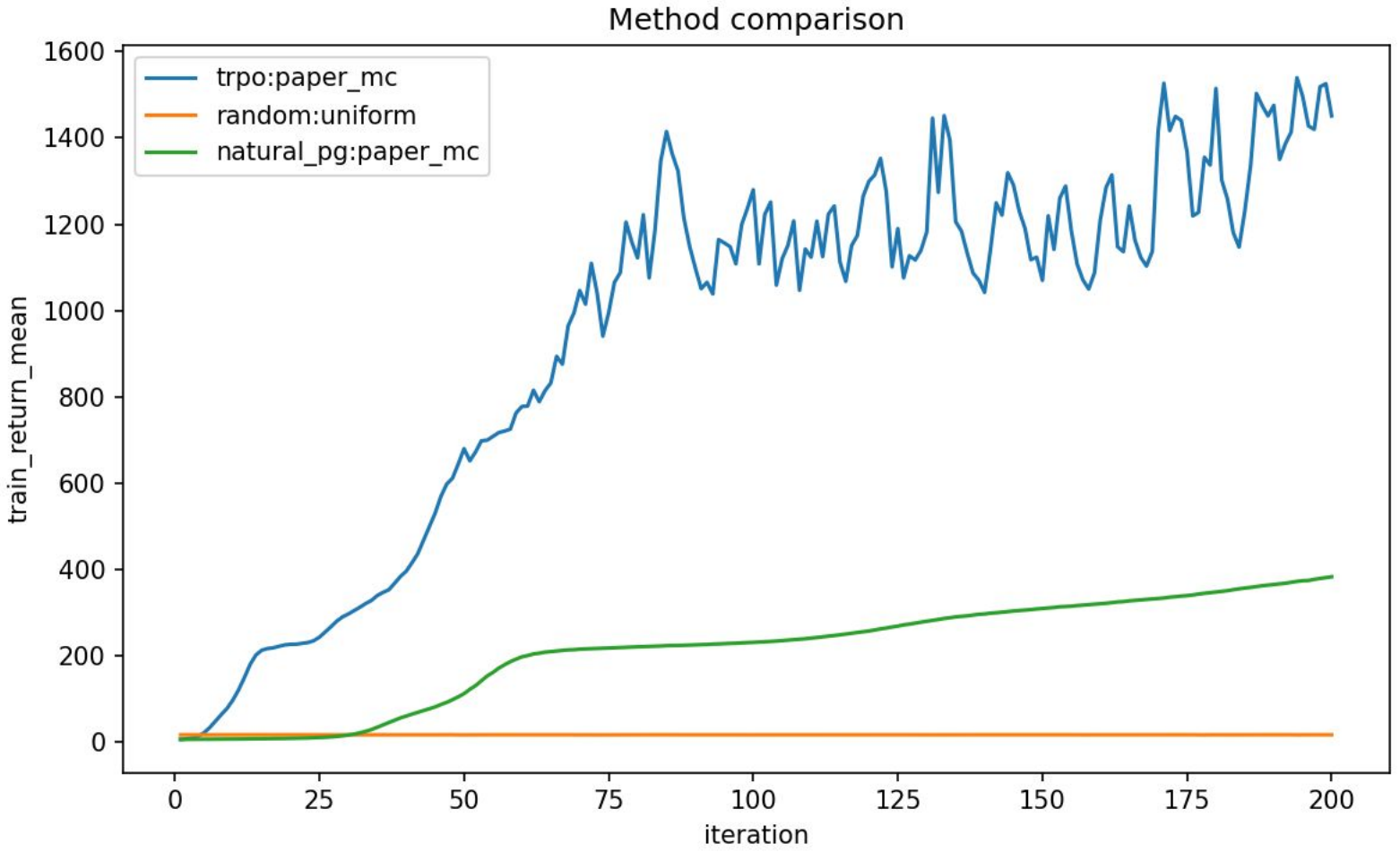
- We performed simulations for the swimmer and the hopper using TRPO and NPG. See figure[ref 2].
- We used the mujoco simulator for these locomotions tasks.

Swimmer



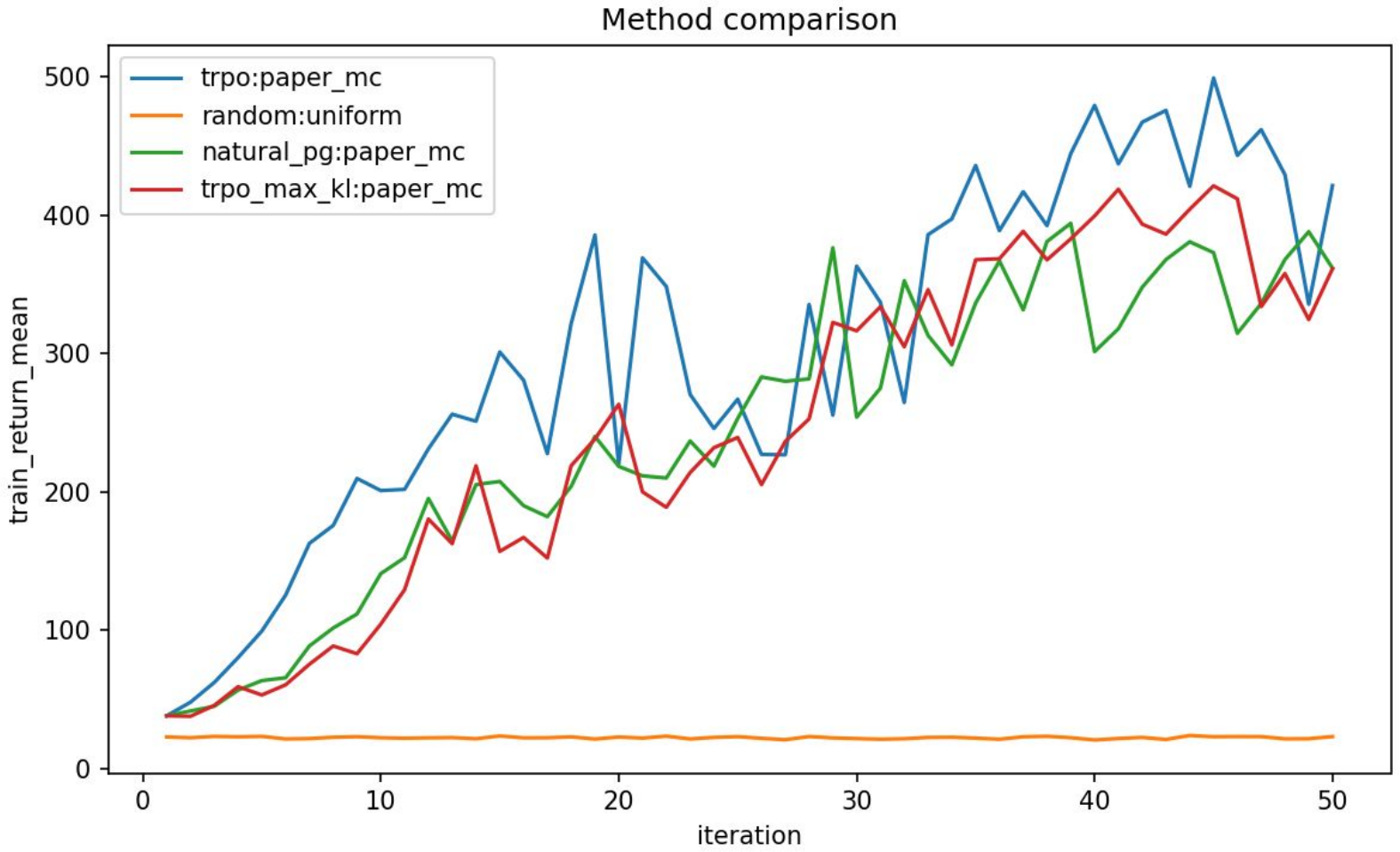
- using mujoco simulator
- ran for 1 epoch

Hopper



- TRPO outperformed NPG significantly.
- Empirical evidence that constraining the KL divergence is a more robust way to choose step sizes and make fast, consistent progress, compared to using a fixed penalty.

Cartpole



- For the lightweight task for cartpole case, we also were able to compute the max kl trpo method,
- This gave similar results, helping to justify our assumption to relax the max kl to mean kl.

Conclusions

- Putting constraints on gradient based updates made policy optimization more reliable.
- CPI, NPG, and TRPO provide a theoretical foundation for safe and structured policy improvement.
- TRPO's lasting contribution is the trust-region perspective that directly influenced later methods such as PPO.
- A future direction is to understand how PPO retains this idea while making the updates simpler and more practical. Also perform more comprehensive empirical studies.

References

1. “Policy gradient methods for reinforcement learning with function approximation,” Sutton et al, 1999
2. “Trust Region Policy Optimization,” Schulman et al. 2015
3. “A Natural Policy Gradient,” Sham Kakade, 2001
4. “Approximately Optimal Approximate Reinforcement Learning,” Kakade and Langford, 2002
5. “Proximal Policy Optimization,” Schulman et al, 2017
6. Tengyang Xie, 2026, https://mdp.sh/teaching/CS839-Spring26/lec9.pdf
7. Tengyang Xie, 2026, https://mdp.sh/teaching/CS839-Spring26/lec10.pdf
8. Advanced policy gradients: natural gradient and TRPO (Schulman) https://rail.eecs.berkeley.edu/deeprlcoursesp17/docs/lec5.pdf
9. Joshua Achiam, https://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_13_advanced_pg.pdf

Thank you!

Questions...