Trust Region Policy Optimization

ISPR Midterm 4

Irene Dovichi

19/06/24

Introduction

The Trust Region Policy Optimization paper presents an iterative procedure for optimizing policies that guarantees a monotonic improvement.

The authors extend the study of Kakade and Langford and develop a practical algorithm called TRPO, capable of optimizing complex, nonlinear policies with tens of thousands of parameters. This has been demonstrated in experiments involving tasks such as swimming, hopping, walking, and playing Atari games directly from raw images.

Mathematical Framework. We will consider a MDP $(S, A, P, r, \rho_0, \gamma)$ and a stochastic policy $\pi : S \times A \to [0,1]$.

Our goal is to maximize the expected discounted reward:

$$\eta(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right]$$

where $s_0 \sim \rho_0(s_0),\ a_t \sim \pi(a_t|s_t),\ s_{t+1} \sim P(s_{t+1}|s_t, a_t)$.

Policy Optimization. We are therefore interested in understanding whether a policy update $\pi \to \tilde{\pi}$ increases $\eta(\tilde{\pi})$ compared to $\eta(\pi)$. It holds that:

$$\eta(\tilde{\pi}) = \eta(\pi) + \sum_s \underbrace{\boxed{\rho_{\tilde{\pi}}(s)}}_{\text{(unnormalized) discounted visitation frequencies}} \sum_a \tilde{\pi}(a|s) \underbrace{\boxed{A_\pi(s, a)}}_{\text{advantage function}} \tag{1}$$

Model Description

The complex dependency of $\rho_{\tilde{\pi}}(s)$ on $\tilde{\pi}$ complicates the direct optimization of Equation (1). We then move on to consider a **local approximation** of $\eta(\tilde{\pi})$:

$$L_\pi(\tilde{\pi}) = \eta(\pi) + \sum_s \boxed{\rho_\pi(s)} \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

the only difference compared to (1)

Kakade and Langford provide a direct relation between $L_\pi(\tilde{\pi})$ and $\eta(\tilde{\pi})$ for **mixture policies** $\tilde{\pi}$. This policy class is restrictive in practice, and the authors of the TRPO paper extend it to **general stochastic policies** obtaining that:

$$\eta(\tilde{\pi}) \geq \boxed{L_\pi(\tilde{\pi}) - C \boxed{D_{KL}^{\max}(\pi, \tilde{\pi})}}, \quad \text{where } C = \frac{4\boxed{\varepsilon}\gamma}{(1-\gamma)^2} \qquad (2)$$

Kullback-Leibler divergence

$$D_{KL}^{\max}(\pi, \tilde{\pi}) = \max_s \boxed{D_{KL}}(\pi(\cdot|s) \,||\, \tilde{\pi}(\cdot|s))$$

$$\varepsilon = \max_{s,a} |A_\pi(s,a)|$$

In the next slide we will see an Algorithm that implements a policy iteration using the **policy improvement bound** of Equation (2), assuming exact A_π evaluation and ensuring monotonically improving policies.

Policy Iteration Algorithm

We report the iteration step directly for the **parametrized policies** $\pi_\theta(a|s)$, as they are the ones that are considered in practice.

To simplify the **notation**, we impose: $L_\theta(\tilde{\theta}) = L_{\pi_\theta}(\pi_{\tilde{\theta}}),\ D_{KL}(\theta \,||\, \tilde{\theta}) = D_{KL}(\pi_\theta \,||\, \pi_{\tilde{\theta}})$.

<u>**Policy Iteration Algorithm**</u>:

- Initialize π_{θ_0}
- For $i = 0, 1, \dots$ until convergence do: $\theta_{i+1} \leftarrow \underset{\theta}{\operatorname{argmax}} \boxed{\left[L_{\theta_i}(\theta) - C\, D_{KL}^{\max}(\theta_i, \theta)\right]}$

Problem. In practice, if we used the large penalty coefficient C the algorithm would be **inefficient** because we would have very small steps.

Solution. To address this issue, the authors propose to **constrain** the KL divergence instead of applying the penalty to it. Additionally, they recommend using the **mean** of the KL divergence instead of the maximum value, as the latter introduces a large number of constraints, making the problem impractical to solve.

$$\bar{D}_{\mathrm{KL}}^{\rho}(\theta_1, \theta_2) = \mathbb{E}_{s\sim\rho}\left[D_{KL}\left(\pi_{\theta_1}(\cdot|s) \,||\, \pi_{\theta_2}(\cdot|s)\right)\right]$$

Solving the Constrained Problem

The **optimization problem** that we have to solve in order to generate a policy update is the following:

$$\boxed{\begin{aligned} &\max_{\theta} \; L_{\theta_{\text{old}}}(\theta) \\ &\text{s.t.} \; \bar{D}_{\text{KL}}^{\rho_{\theta_{\text{old}}}}(\theta_{\text{old}}, \theta) \leq \boxed{\delta} \end{aligned}} \tag{3}$$

fixed hyperparameter

Making a linear approximation to the objective *L* and a quadratic approximation to the KL term we get:

$$g = \frac{\partial L(\theta)}{\partial \theta}\Big|_{\theta=\theta_{\text{old}}}$$

$$\boxed{\begin{aligned} &\max_{\theta} \; \boxed{g}(\theta - \theta_{\text{old}}) \\ &\text{s.t.} \; \frac{1}{2}(\theta - \theta_{\text{old}})^T \boxed{H}(\theta - \theta_{\text{old}}) \leq \delta \end{aligned}}$$

$$H = \frac{\partial^2 \bar{D}_{\text{KL}}(\theta_{\text{old}}, \theta)}{\partial^2 \theta}\Big|_{\theta=\theta_{\text{old}}}$$

We differentiate w.r.t. θ the Lagrangian $\mathcal{L}(\theta, \lambda) = g(\theta - \theta_{\text{old}}) - \frac{\lambda}{2}\left[(\theta - \theta_{\text{old}})^T H(\theta - \theta_{\text{old}}) - \delta\right]$ and we get that:

scaling parameter β s.t. the KL constraint holds with equality

$$\theta - \theta_{\text{old}} = \boxed{\sqrt{\frac{2\delta}{(\frac{1}{\lambda}H^{-1}g)^T H(\frac{1}{\lambda}H^{-1}g)}}}\frac{1}{\lambda}H^{-1}g$$

Notice that $H^{-1}g$ can be computed without forming the full Hessian, using the **conjugate gradient** algorithm.

Practical Algorithm

Finally, we use a **Monte Carlo simulation** to approximate both the objective and the constraint functions. With some mathematical manipulation, we rewrite Problem (3) as:

$$\begin{aligned}\max_{\theta} \quad & \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}, a\sim q}\left[\frac{\pi_\theta(a|s)}{\boxed{q(a|s)}}\boxed{Q_{\theta_{\text{old}}}(s,a)}\right] \\ \text{s.t.} \quad & \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{KL}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\,||\,\pi_\theta(\cdot|s)\right)\right] \le \delta\end{aligned} \tag{4}$$

state-action value function

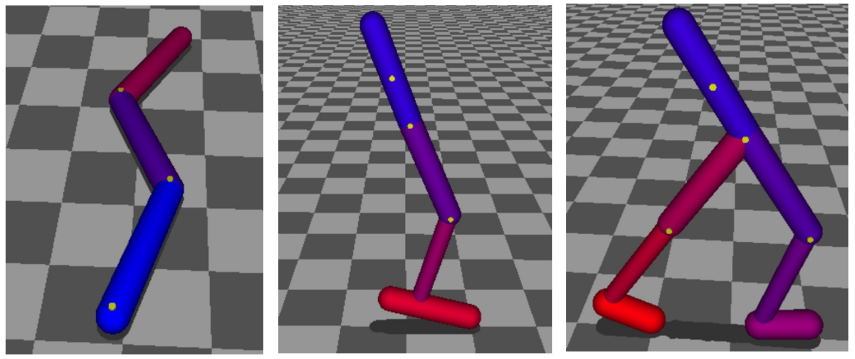
sampling distribution

<u>Practical Algorithm</u>:

- Use **single path** or **vine** procedures to collect state-action pairs and their Monte Carlo Q-value estimates
- Get an estimation of the objective and constraint functions in (4) by averaging over samples
- Approximately solve the constrained optimization problem using **CG** to compute the search direction $H^{-1}g$ followed by a **line search**: start with the maximal value of the step length β $\left(\theta - \theta_{old} = {}^{\beta}/_{\gamma}\, H^{-1}g\right)$ and shrink it until the objective improves, getting $\theta - \theta_{old} = \alpha\, H^{-1}g$
- Update the policy parameter: $\theta \leftarrow \theta_{old} + \alpha\, H^{-1}g$

Experiments

The experiments were conducted using the MuJoCo simulator for **robotic locomotion**, evaluating models such as the *swimmer*, the *hopper*, the *walker*, and the classic *cart-pole* balancing problem as a baseline.



- The comparison included various algorithms, and among them single path TRPO and vine TRPO achieved the **best results**, solving all problems and yielding the best solutions.
- The heuristic approximation that considers $\overline{D}_{KL}$ instead of D_{KL}^{max} showed overall similar results in less time.

Additionally, TRPO was evaluated in the context of playing **Atari games** using raw images as input.

- TRPO achieved reasonable scores across most games, and in some cases outperformed other tested methods. In the Enduro game (Figure on the right) it surpassed them all.
- Unlike prior methods, the TRPO approach was not specifically designed for this task, demonstrating its **versatility**.

Final Considerations

Novelties

- TRPO switches from unconstrained to constrained optimization by using a constrain on the KL divergence instead of a penalty coefficient, allowing robust and controlled policy updates that ensure significant progress without destabilizing the learning.
- In the field of robotic locomotion, they effectively developed controllers for swimming, hopping, and walking using a generic policy search method and non-engineered, general-purpose policy representations.

Strong points

- TRPO is powerful because it extends an already existing algorithm that guarantees a non-decreasing expected return η to a broader class of policies that are of practical interest: general stochastic policies.
- Good empirical results on a range of challenging policy learning tasks, outperforming prior methods.

Weaknesses

- The vine method has the advantage of providing much lower variance in the local estimate of the objective than the single path. However, it requires many more simulator calls and multiple trajectories from each state, limiting its use to systems that can be reset to arbitrary states.
- High computational cost, particularly due to the need for the inverse Hessian matrix of the KL divergence.