Understanding "Trust Region Policy Optimization"

Ziyue Wang Yu Zhang
Qishi Journal Club

March 20, 2022

Overview

1. Theory
   - Motivation
   - TRPO Steps
     - Minorize-Maximization (MM)
     - Natural Gradient
   - TRPO Algo: TNPG + line search
   - TRPO Limitation
   - Remedy: Proximal Policy Optimization
2. Implementation
   - Benchmarks
3. Summary

Motivation

- Policy optimization categories: policy iterations, policy gradient, derivative-free optimization (CEM, CMA[1])
  - "Gradient-based methods performs worse than gradient-free random search is unsatisfying, since gradient-based optimization algorithms enjoy much better **sample complexity** guarantees than gradient-free methods."
  - "Extending gradient-based optimization success to reinforcement learning would allow for efficient training of **complex and powerful policies**."
- Distance in parameter space $\neq$ distance in policy space
  - sample inefficiency
  - unstable training: hard to find a good learning rate
- "Search for the biggest parameter change $\nabla\theta$ inducing the smallest change in the policy, but in the right direction"

[1] Cross Entropy Method, Covariance Matrix Adaptation

TRPO Steps

1. Equation

$$\max_{\hat{\theta}} \eta(\hat{\theta}) = \max_{\hat{\theta}} \eta(\hat{\theta}) - \eta(\theta)$$
$$\rightarrow \text{(MM)} \max_{\hat{\theta}} \mathcal{L}_\theta(\hat{\theta}) - C\bar{D}^{\rho}_{KL}(\theta, \hat{\theta})$$
$$\leftrightarrow \max_{\hat{\theta}} \mathcal{L}_\theta(\hat{\theta}), s.t. \bar{D}^{\rho}_{KL}(\theta, \hat{\theta}) \leq \delta$$

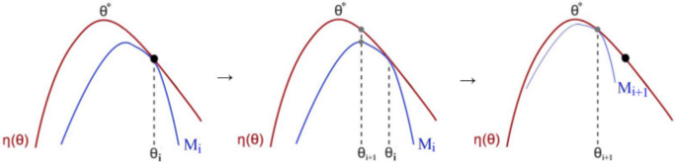
- Local Approximation η
  - $\eta(\hat{\theta}) = \eta(\theta) + \sum_s \rho_{\hat{\theta}}(s) \sum_a \hat{\theta}(a|s) A_\theta(s, a)$
  - Surrogate objective function
    $\mathcal{L}_\theta(\hat{\theta}) = \eta(\theta) + \sum_s \rho_{\theta}(s) \sum_a \hat{\theta}(a|s) A_\theta(s, a)$
  - First Order Approx. $\mathcal{L}_\theta(\theta) = \eta(\theta)$, $\nabla_\theta \mathcal{L}_\theta(\hat{\theta})|_{\hat{\theta}=\theta} = \nabla_\theta \eta(\hat{\theta})|_{\hat{\theta}=\theta}$
  - Discounted Visitation Frequencies
    $\rho_\theta(s) = P_\theta(s_0 = s) + \gamma P_\theta(s_1 = s) + \gamma^2 P_\theta(s_2 = s) + \ldots$
- Why constraint? hard to robustly choose the penalty coefficient C.

TRPO Steps

2. Solve it using truncated (fixed-iteration CG) natural policy gradient with line search
   - line search: verify the new policy first before commits the change to reduce the quadratic approximation error and make sure the trust region δ can indeed improve performance: $\bar{D}_{KL} \leq \delta$ and $\mathcal{L}(\theta) \geq 0$. If the verification fails, we will decay the natural policy gradient by a factor of $\alpha(0 \sim 1)$ until the new parameters meet the requirements above.

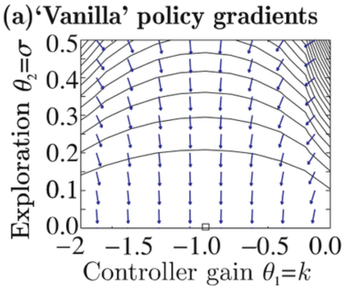
Minorize-Maximization

- **Idea**: iteratively maximize a lower bound function to approximate the expected reward locally. Usually, the lower bound function is easier to optimize than η



- **Steps**:
  1. initial policy guess
  2. find a lower bound M that approximates the expected reward η locally at the current guess
  3. locate the optimal point for M and use it for the next guess
  4. repeat 2 - 3

Natural Gradient



Natural Gradient - Metrics

- Euclidian metric

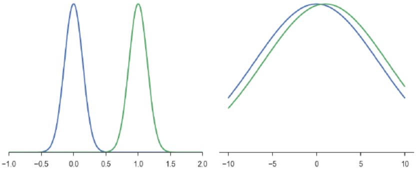
$$||\theta + \Delta\theta - \theta||^2$$

- Riemannian metric

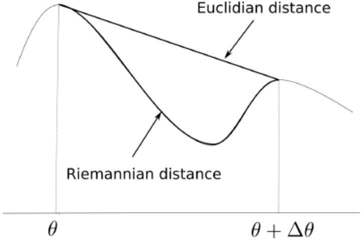
$$< \Delta\theta, F(\theta)\Delta\theta >$$

- KL Divergence

$$D_{KL}(p||q) = \mathbb{E}_{x\sim p}[\log \frac{p(x)}{q(x)}]$$



(a) $N(0, 0.2), N(1, 0.2)$ *vs.* $N(0, 10), N(1, 10)$



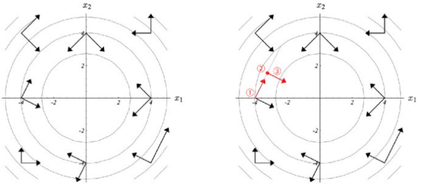
(b) Euclidian vs. Riemmanian

Natural Gradient - Step

1. **'Vanilla' Gradient Ascent Step**: αg, learning rate α, policy gradient $g = \nabla_{\hat{\theta}} \mathcal{L}_\theta(\hat{\theta})$
2. **Natural Gradient Ascent Step**: $\sqrt{\frac{2\delta}{g^T F(\theta)^{-1} g}} F(\theta)^{-1} g$
   - $$\max_{\hat{\theta}} \mathcal{L}_\theta(\hat{\theta}), s.t. D_{KL}(p_\theta || p_{\theta+\Delta\theta}) \leq \delta$$
     $$\max_{\hat{\theta}} \mathcal{L}_\theta(\hat{\theta}) - \lambda(D_{KL}(p_\theta || p_{\theta+\Delta\theta}) - \delta)$$
   - Approximate by using Taylor expansion
     $$\mathcal{L}_\theta(\hat{\theta}) \approx g\Delta\theta$$
     $$D_{KL}(p_\theta || p_{\theta+\Delta\theta}) \approx \Delta\theta^T F(\theta) \Delta\theta$$
   - Fisher Information Matrix
     $$F(\theta) = \nabla^2 D_{KL}(p(x;\theta) || p(x;\theta+\Delta\theta)|_{\Delta\theta=0}$$
     $$\text{Simpler Form } F(\theta) = \mathbb{E}_{x\sim p(x,\theta)}[\nabla log(p;\theta) \nabla log(p;\theta)^T]$$

Natural Gradient - Pros Cons

- **Faster Converges**: inverse of the Fisher information matrix can bring invariance to model parameterization.
- **Inaccuracy and slowness**: inverse of Fisher Matrix. Partially solved it by using Conjugate Gradients: a line search method to solve $Ax = b \rightarrow x = A^{-1}b$ without undoing part of the moves done previously. It optimizes a quadratic equation in fewer step than the gradient ascent.
  1. start the ascent in one particular direction
  2. settle down in the optimal point for that direction
  3. find a new direction d_j which is conjugate to any previous moving d_i



TRPO Algo: TNPG + line search

Input: initial policy parameters θ;

for $k = 0, 1, 2, \dots$ **do**

Collect set of trajectories D_k on policy $\pi_k = \pi(\theta_0)$;

Estimate advantages $\hat{A}_{(}\theta_0)$;

Form sample estimate for;

- policy gradient $\hat{g}_k$
- and KL-divergence Hessian-vector product $f(v) = \hat{F}_k v$

Use CG with n_{cg} iterations to obtain $x_k \approx \hat{F}_k^{-1}\hat{g}_k$;

Estimate proposed step $\Delta_k = \sqrt{\frac{2\delta}{\hat{g}_k^T \hat{F}(\theta)^{-1}\hat{g}_k}}\hat{F}(\theta_k)^{-1}\hat{g}_k$;

Perform backtracking line search with exponential decay to obtain final update

$$\theta_{k+1} = \theta_k + \alpha^j \Delta_k$$

end

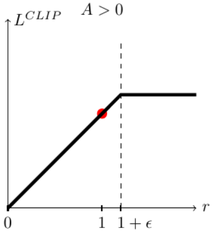
Algorithm 1: Trust Region Policy Optimization

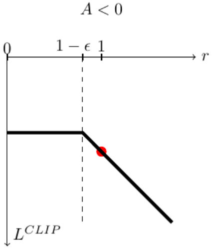
TRPO Limitation

1. Scalability: computing F is expensive, which requires a large batch of rollouts to approximate it accurately.
2. Complexity: conjugate gradient is more complicated and less flexible than SGD

Remedy: PPO

- **Clip the estimated advantage function** by using the difference between the new and old policy
  - $r_t(\hat{\theta}) = \frac{\hat{\theta}(a_t|s_t)}{\theta(a_t|s_t)}, r_t(\theta) = 1$
  - conservative policy iteration $\mathcal{L}_{\theta}^{CPI}(\hat{\theta}) = \mathbb{E}_t[r_t(\hat{\theta})\hat{A}_t^{\theta}]$
  - $\mathcal{L}_{\theta}^{CLIP}(\hat{\theta}) = \mathbb{E}_t[\min(r_t(\hat{\theta})\hat{A}_t^{\theta}, clip(r_t(\hat{\theta}), 1-\epsilon, 1+\epsilon)\hat{A}_t^{\theta})]$, where $\epsilon = 0.2$.
- **Adaptive KL Penalty** (instead of constraint) to improve the speed closer to the gradient descent method





Benchmarks

- Package: Tianshou, fast and comprehensive
- Caveat: Hard to reproduce RL results in general. link
- Benchmarks for MOJUCO environments

Summary

1. Theory: **TRPO** uses **MM** to guarantee a steady policy improvement. It applies **truncated natural policy gradient** and **line search** method to update the policy. To solve the **heavy computation issue** regarding H matrix, **PPO** method achieves a good balance between speed and error rate by using Advantage Function Clipping and Adaptive KL Penalty.
2. Implementation: Tianshou package provides a **fast** speed and **comprehensive** comparison between different RL algos.

References

1. Paper "Trust Region Policy Optimization"
2. Paper "Proximal Policy Optimization"
3. Paper "On Policy Gradient"
4. deeprl notes from Julien
5. Depth First Learning Curriculum - TRPO
6. Medium Meterial
7. Tianshou Github Repository
8. Natural Gradient