Issues in Policy Gradient

Insoon Yang

Department of Electrical and Computer Engineering
Seoul National University

Control + Optimization Research Lab

Review: Parameterized policy optimization

- Parameterization of policy:

$$\pi_\theta(a|s),$$

where $\theta \in \mathbb{R}^l$ is a parameter vector (weights)

Review: Parameterized policy optimization

- Parameterization of policy:

$$\pi_\theta(a|s),$$

where $\theta \in \mathbb{R}^l$ is a parameter vector (weights)

- The problem of finding the best parameters:

$$\max_\theta \; J(\theta) := \mathbb{E}^{\pi_\theta}\left[\sum_t r(s_t, a_t)\right]$$

Review: Policy gradient

- The problem of finding the best policy parameters::

$$\max_{\theta} \; J(\theta) := \mathbb{E}^{\pi_\theta}\left[\sum_t r(s_t, a_t)\right]$$

Review: Policy gradient

- The problem of finding the best policy parameters::

$$\max_{\theta} \; J(\theta) := \mathbb{E}^{\pi_\theta}\left[\sum_t r(s_t, a_t)\right]$$

- Policy gradient method:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta),$$

where α is the stepsize and

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) A(s_t^i, a_t^i)$$

Challenge I: Good direction to move θ?

Policy gradient: Moving in the direction of the gradient $\nabla_\theta J(\theta)$:

$$\theta \leftarrow \theta + \underbrace{\alpha}_{stepsize} \nabla_\theta J(\theta)$$

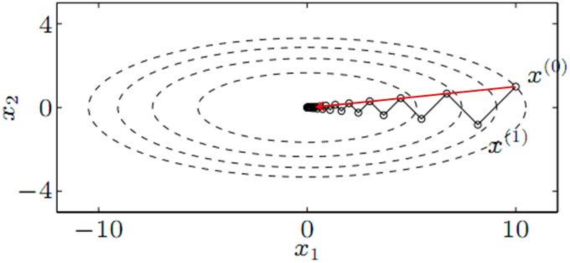
Challenge I: Good direction to move θ?

Policy gradient: Moving in the direction of the gradient $\nabla_\theta J(\theta)$:

$$\theta \leftarrow \theta + \underbrace{\alpha}_{stepsize} \nabla_\theta J(\theta)$$

zigzag < - first order 만 들어가기 때문에,
Hessian matrix같은 second order derevatives를 이용하면 효율적일 수 있음 (Newton - rapson or Conjugated gradients)

Q) Can we do better?



Challenge II: Choice of Stepsize

Gradient ascent:

$$\theta \leftarrow \theta + \underbrace{\alpha}_{stepsize} \nabla_\theta J(\theta)$$

- Too large stepsize: Can be a bad move
- Too small stepsize: Slow learning speed

Q) Why?

Challenge III: Sample inefficiency

REINFORCE algorithm:

1. Sample $\{\tau^i\} := \{(s_0, a_0, \ldots, s_T, a_T)\}$ using the **current policy** $\pi_\theta(a_t|s_t)$

2. Estimate the gradient

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \right) \left(\sum_{t=0}^{T} r(s_t^i, a_t^i) \right)$$

3. Perform gradient ascent:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta);$$

On-policy:

- new samples to be generated after each update of policy

Summary of challenges in policy gradient

- Unclear if $\nabla_\theta J$ is the best direction to move
- Unclear how to choose stepsizes α
- Poor sample efficiency

How can we resolve the challenges?

How can we resolve the challenges?

Trust region policy optimization (TRPO)

Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Abstract

In this article, we describe a method for optimizing control policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified scheme, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is effective for optimizing large nonlinear policies such as neural networks. Our experiments demonstrate its robust performance on a wide variety of tasks: learning simulated robotic swimming, hopping, and walking gaits; and playing Atari games using images of the screen as input. Despite its approximations that deviate from the theory, TRPO tends to give monotonic improvement, with little tuning of hyperparameters.

namic programming (ADP) methods, stochastic optimization methods are difficult to beat on this task (Gabillon et al., 2013). For continuous control problems, methods like CMA have been successful at learning control policies for challenging tasks like locomotion when provided with hand-engineered policy classes with low-dimensional parameterizations (Wampler & Popović, 2009). The inability of ADP and gradient-based methods to consistently beat gradient-free random search is unsatisfying, since gradient-based optimization algorithms enjoy much better sample complexity guarantees than gradient-free methods (Nemirovski, 2005). Continuous gradient-based optimization has been very successful at learning function approximators for supervised learning tasks with huge numbers of parameters, and extending their success to reinforcement learning would allow for efficient training of complex and powerful policies.

How can we resolve the challenges?

How TRPO resolves the challenges

How can we resolve the challenges?

How TRPO resolves the challenges

- Unclear if $\nabla_\theta J$ is the best direction to move:
  Use a surrogate objective to guarantee monotonic improvement

How can we resolve the challenges?

How TRPO resolves the challenges

- Unclear if $\nabla_\theta J$ is the best direction to move:
  Use a surrogate objective to guarantee monotonic improvement

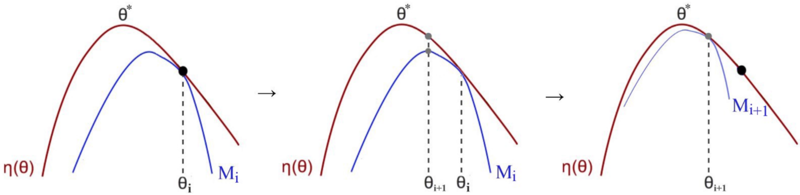
- Unclear how to choose stepsizes α:
  Take a large step as far as we can trust (trust region constraint)

How can we resolve the challenges?

How TRPO resolves the challenges

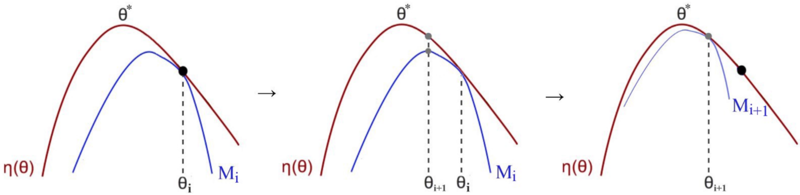
- Unclear if $\nabla_\theta J$ is the best direction to move:
  Use a surrogate objective to guarantee monotonic improvement
- Unclear how to choose stepsizes α:
  Take a large step as far as we can trust (trust region constraint)
- Poor sample efficiency:
  Use importance sampling

TRPO Basics I: Minorize-Maximization (MM) algorithm



To find the maximum of the red curve η:

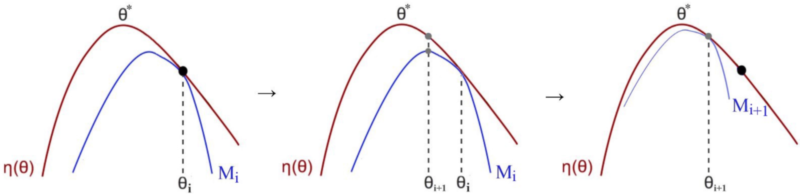
TRPO Basics I: Minorize-Maximization (MM) algorithm



To find the maximum of the red curve η:

1. Construct a curve M_i (blue) that lower bounds η such that $M_i(\theta_i) = \eta(\theta_i)$;

TRPO Basics I: Minorize-Maximization (MM) algorithm

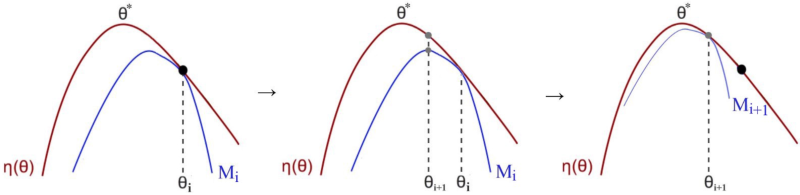


To find the maximum of the red curve η:

1. Construct a curve M_i (blue) that lower bounds η such that $M_i(\theta_i) = \eta(\theta_i)$;
2. Find the maximum of M_i: θ_{i+1}

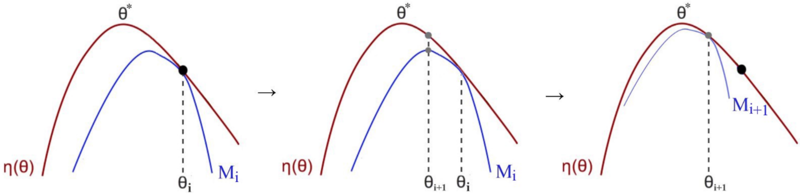
Repeat these until convergence

Surrogate in MM algorithm



The lower-bound function is called “*Surrogate*”

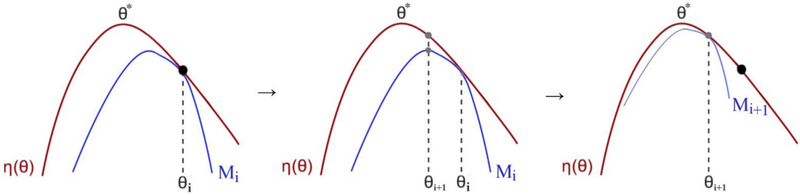
Surrogate in MM algorithm



The lower-bound function is called "*Surrogate*"

Q) How should we choose a surrogate?

Surrogate in MM algorithm

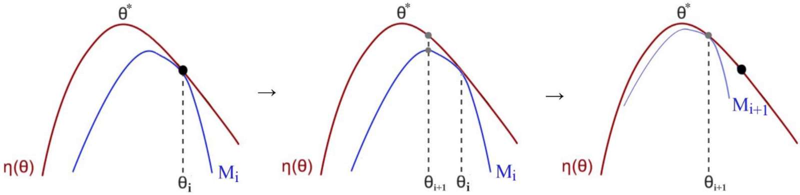


The lower-bound function is called "*Surrogate*"

Q) How should we choose a surrogate?

- Easy to maximize:

Surrogate in MM algorithm

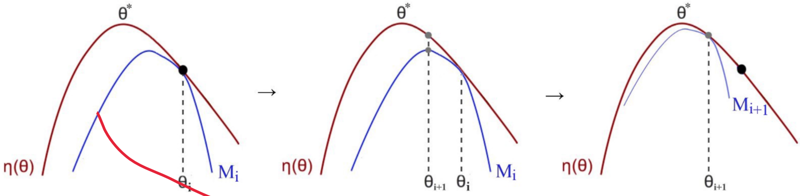


The lower-bound function is called "*Surrogate*"

Q) How should we choose a surrogate?

- Easy to maximize:
- Example: $M(\theta) = c + g^\top \theta + \frac{1}{2}\theta^\top F \theta$

Surrogate in MM algorithm



The lower-bound function is called "*Surrogate*"

Q) How should we choose a surrogate?

- Easy to maximize:
- Example: $M(\theta) = c + g^\top \theta + \frac{1}{2}\theta^\top F \theta$
- Maximum at $\theta^* = -F^{-1}g$

TRPO Basics II: Trust region method

Key idea:

TRPO Basics II: Trust region method

Key idea:

- Determine the maximum stepsize δ that we want to explore

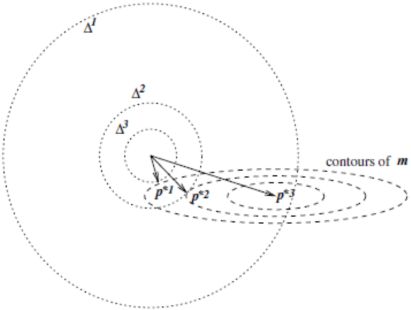
TRPO Basics II: Trust region method

Key idea:

- Determine the maximum stepsize δ that we want to explore
- Find the optimal point within this trust region:

$$\begin{aligned} \max_{\theta \in \mathbb{R}^n} \quad & J_{app}(\theta) \\ \text{s.t.} \quad & \|\theta\| \leq \delta, \end{aligned}$$

where J_{app} is an approximation of the original objective



How to adjust the size of trust region?

How to adjust the size of trust region?

Classical version:

- Expand δ if our approximation J_{app} of J is good
- Shrink δ if our approximation is poor

How to adjust the size of trust region?

Classical version:

- Expand δ if our approximation J_{app} of J is good
- Shrink δ if our approximation is poor

TRPO version:

- Expand δ if our parameterized policy π_θ is changing too little
- Shrink δ if our policy is changing too much

TRPO Basics III: Importance sampling

Suppose we want to calculate

$$\mathbb{E}_{x\sim p}[f(x)],$$

where it is difficult to generate samples according to p

TRPO Basics III: Importance sampling

Suppose we want to calculate

$$\mathbb{E}_{x \sim p}[f(x)],$$

where it is difficult to generate samples according to p

Q) Can we use a different distribution q to generate samples?

TRPO Basics III: Importance sampling

Suppose we want to calculate

$$\mathbb{E}_{x \sim p}[f(x)],$$

where it is difficult to generate samples according to p

Q) Can we use a different distribution q to generate samples?

- Importance sampling:

q로 만든 sample

$$\mathbb{E}_{x \sim q}\left[\frac{p(x)}{q(x)} f(x)\right] \approx \frac{1}{N} \sum_i \underbrace{\frac{p(x^i)}{q(x^i)}}_{\text{calibration}} f(x^i),$$

ex) gaussian

where $\{x^i\} \sim q$.

Application of importance sampling to policy gradient

Application of importance sampling to policy gradient

Instead of using π_θ, we use π_{old} to generate samples

Application of importance sampling to policy gradient

Instead of using π_θ, we use π_{old} to generate samples

- Objective: $J(\theta) = \mathbb{E}_{\tau \sim p_\theta}[r(\tau)] = \mathbb{E}_{\tau \sim p_{old}}[\frac{p_\theta(\tau)}{p_{old}(\tau)} r(\tau)]$

Application of importance sampling to policy gradient

Instead of using π_θ, we use π_{old} to generate samples

- Objective: $J(\theta) = \mathbb{E}_{\tau \sim p_\theta}[r(\tau)] = \mathbb{E}_{\tau \sim p_{old}}[\frac{p_\theta(\tau)}{p_{old}(\tau)} r(\tau)]$
- Calculate the ratio:

$$\frac{p_\theta(\tau)}{p_{old}(\tau)} = \frac{p(s_0)\prod_{t=0}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)}{p(s_0)\prod_{t=0}^{T} \pi_{old}(a_t|s_t) p(s_{t+1}|s_t, a_t)} = \frac{\prod_{t=0}^{T} \pi_\theta(a_t|s_t)}{\prod_{t=0}^{T} \pi_{old}(a_t|s_t)}$$

Application of importance sampling to policy gradient

Instead of using π_θ, we use π_{old} to generate samples

- Objective: $J(\theta) = \mathbb{E}_{\tau \sim p_\theta}[r(\tau)] = \mathbb{E}_{\tau \sim p_{old}}[\frac{p_\theta(\tau)}{p_{old}(\tau)} r(\tau)]$
- Calculate the ratio:

$$\frac{p_\theta(\tau)}{p_{old}(\tau)} = \frac{p(s_0)\prod_{t=0}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)}{p(s_0)\prod_{t=0}^{T}\pi_{old}(a_t|s_t)p(s_{t+1}|s_t,a_t)} = \frac{\prod_{t=0}^{T}\pi_\theta(a_t|s_t)}{\prod_{t=0}^{T}\pi_{old}(a_t|s_t)}$$

- Policy gradient:

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{\tau \sim p_{old}}\left[\frac{p_\theta(\tau)}{p_{old}(\tau)} \nabla_\theta \log p_\theta(\tau) r(\tau)\right] \\ &= \mathbb{E}_{\tau \sim p_{old}}\left[\left(\frac{\prod_{t=0}^{T}\pi_\theta(a_t|s_t)}{\prod_{t=0}^{T}\pi_{old}(a_t|s_t)}\right)\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)\right) r(\tau)\right]\end{aligned}$$

What's next?

- Using these theoretical and algorithmic tools to develop an **advanced policy gradient (actor-critic)** method $\Longrightarrow$ TRPO

What's next?

- Using these theoretical and algorithmic tools to develop an **advanced policy gradient (actor-critic)** method
$\Longrightarrow$ TRPO
- Features of TRPO:
  - Efficient optimization of parameters
  (good direction and stepsize to move)
  - Sample efficient
  (off-policy)
  - Can be modified for a better performance: ACKTR, PPO, etc.