Reinforce Learning

TRPO, PPO

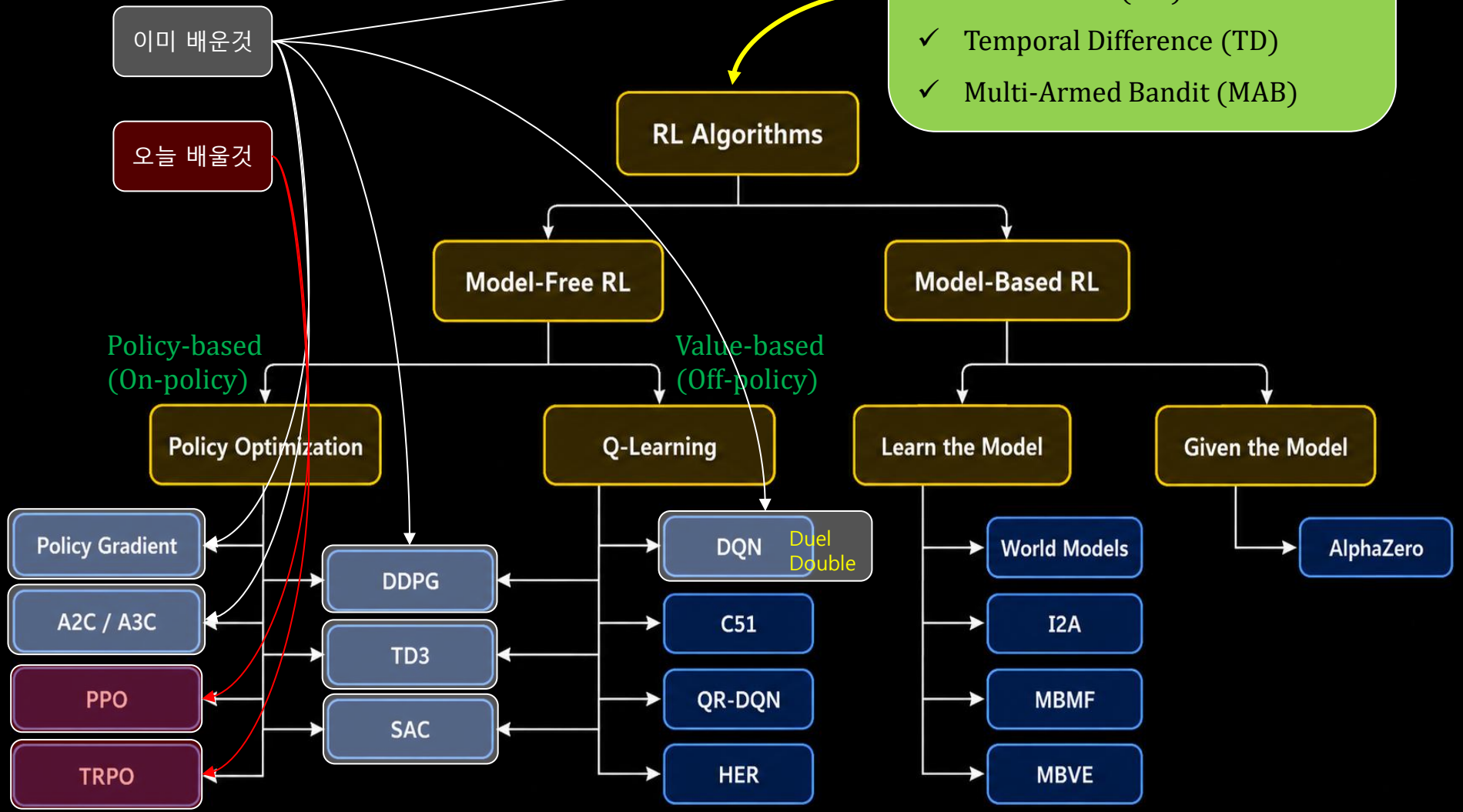
소프트웨어 꼰대 강의

노기섭 교수

(kafa46@hongik.ac.kr)

A Taxonomy of RL Algorithms

- ✓ Markov Decision Process (MDP)
- ✓ Dynamic Programming (DP)
- ✓ Monte Carlo (MC)
- ✓ Temporal Difference (TD)
- ✓ Multi-Armed Bandit (MAB)



Design Principles of Modern RL

Covered these technologies so far!

Function Approximation

Neural networks approximate value and policy functions

Off-policy Learning

Replay buffers improve sample efficiency

Variance Reduction

Advantage estimation reduces gradient variance

Entropy Regularization

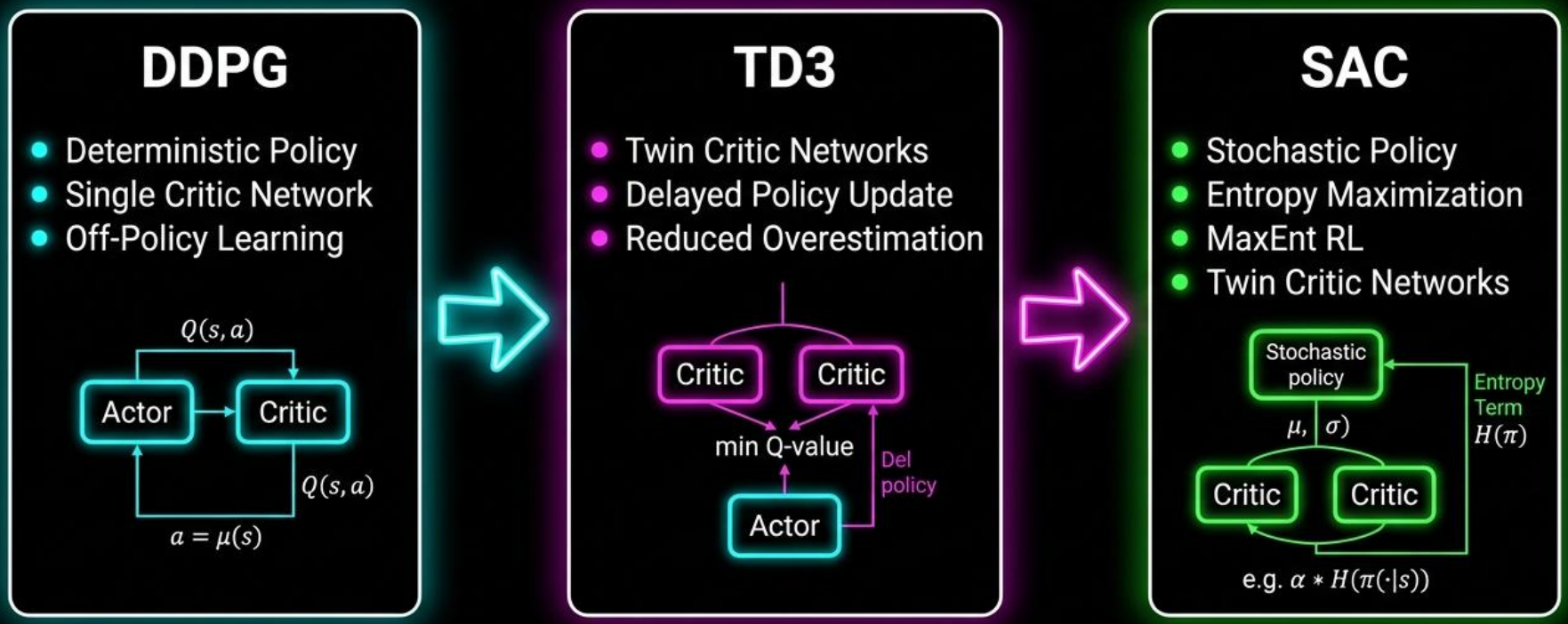
Encourages exploration and prevents premature convergence

We'll explore

Clipped / Constrained Updates

Improve training stability (TRPO, PPO)

Evolution of Continuous Control RL



Motivation: Why Policy Gradient Is Unstable?

Policy Gradient Methods

- Directly optimize the policy
- Can learn complex behaviors
- Effective for continuous action spaces

Need a method that improves policy while preventing excessively large updates.

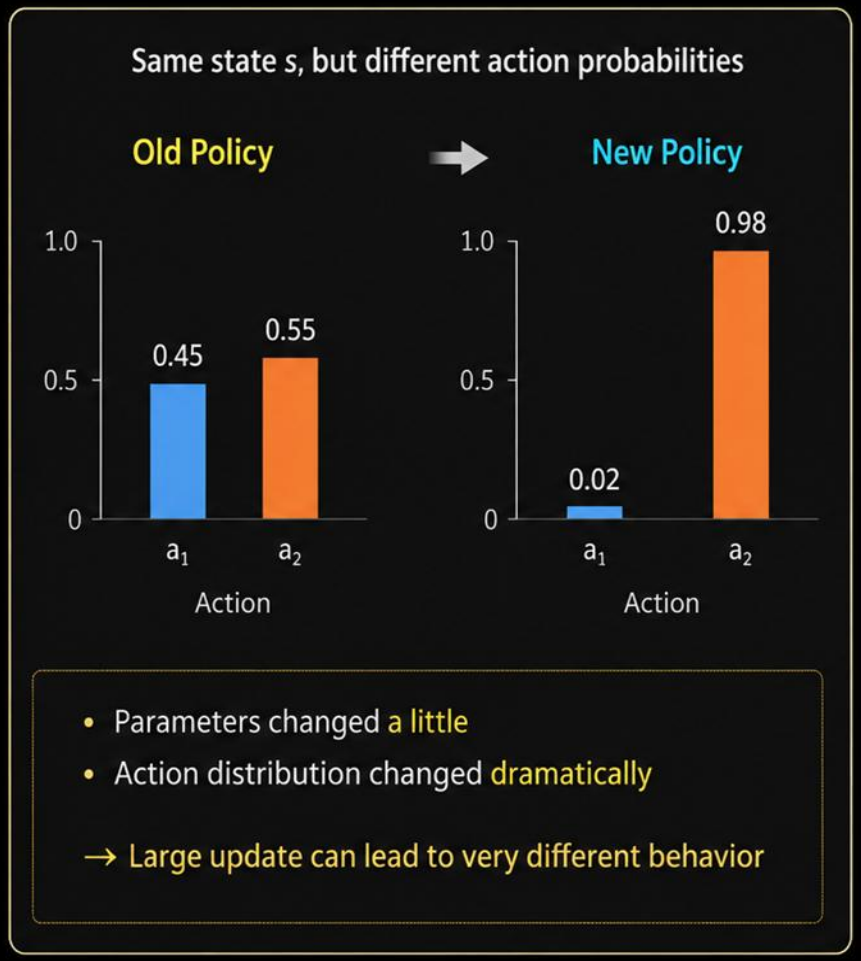
However, …

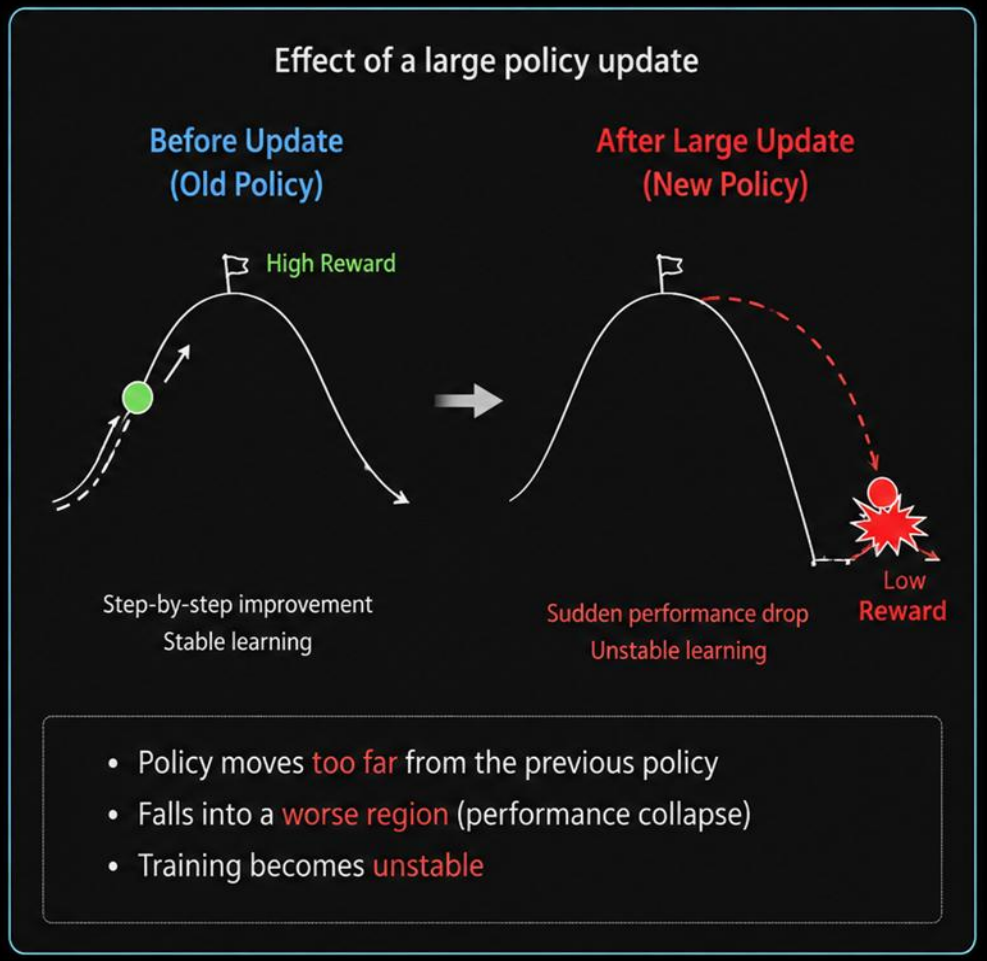
- Large policy updates can be dangerous
- Performance may collapse
- Training can become highly unstable

Why Large Policy Updates Are Dangerous?

Even a small parameter update, policy gradient can drastically change behavior.

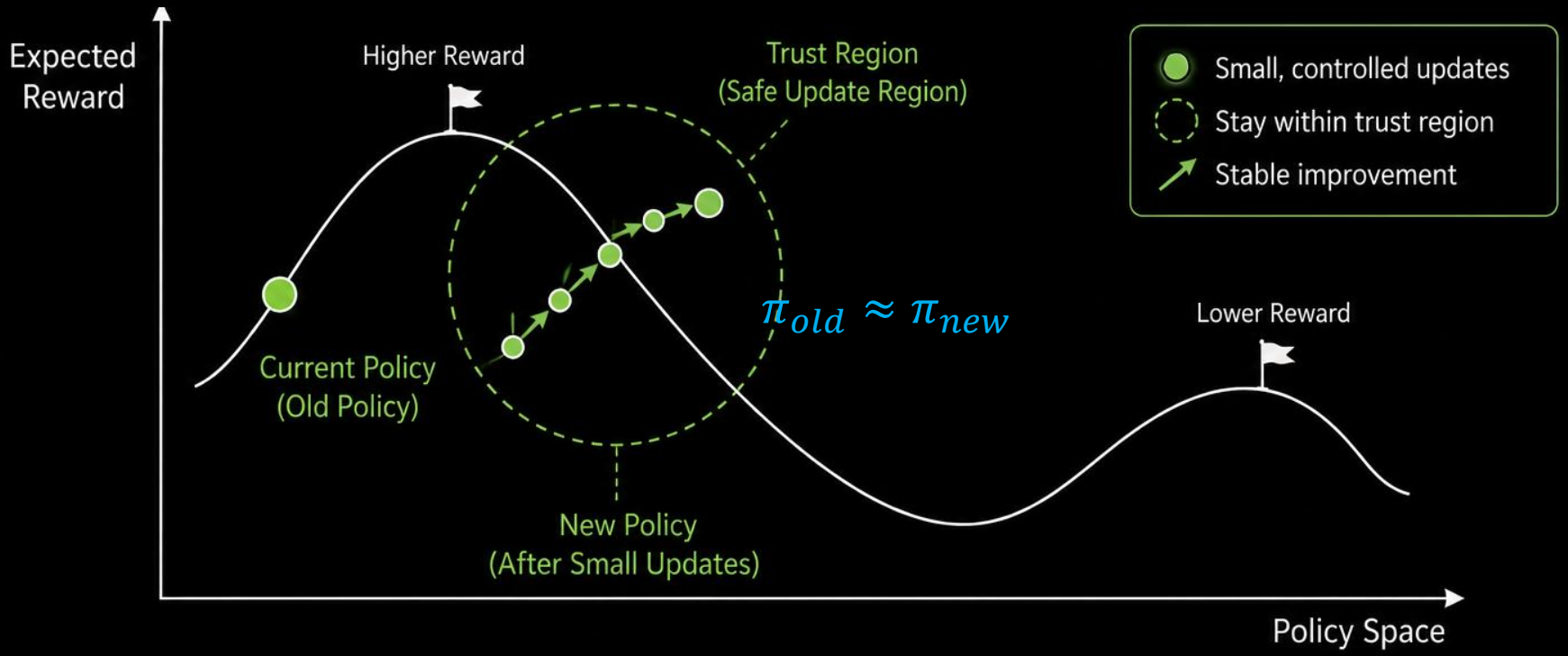
➔ We need a way to limit policy changes safely.





Trust Region Idea

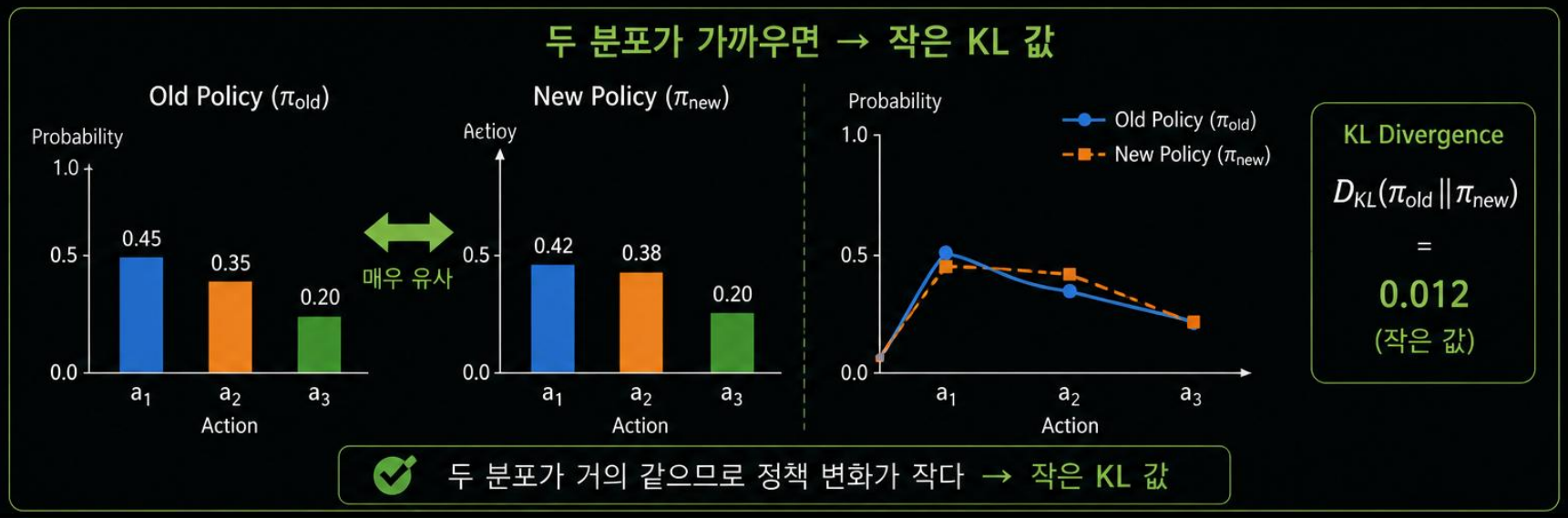
Trust Region:
Restrict policy updates to remain close to the previous policy

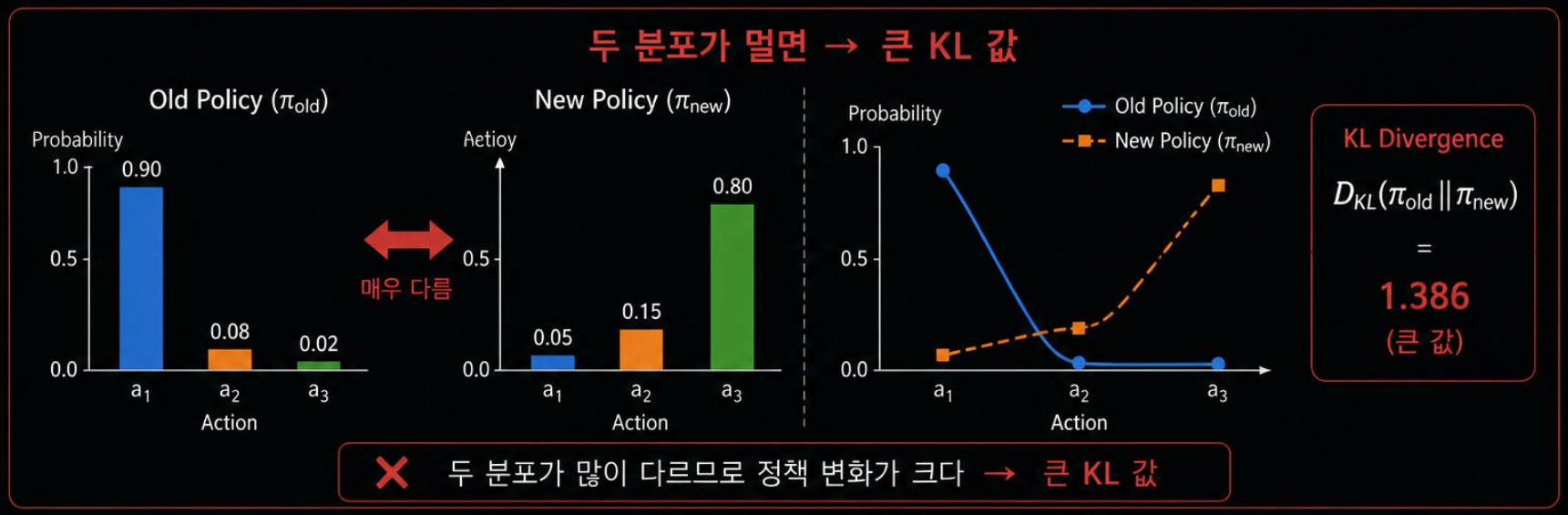


- Small updates keep the policy close to the old policy
- Policy changes smoothly
- Performance improves steadily
- Training remains stable

Stable & Consistent Improvement

Conceptual Representation of KL Divergence in RL





KL Divergence: Distance Between Policies

What is KL Divergence?

- Measures difference between two probability distributions
- Compares old and new policies
- Quantifies policy shift

$$D_{KL}(\pi_{old} || \pi_{new}) \leq \delta$$

Small KL Divergence

→ Similar Policies

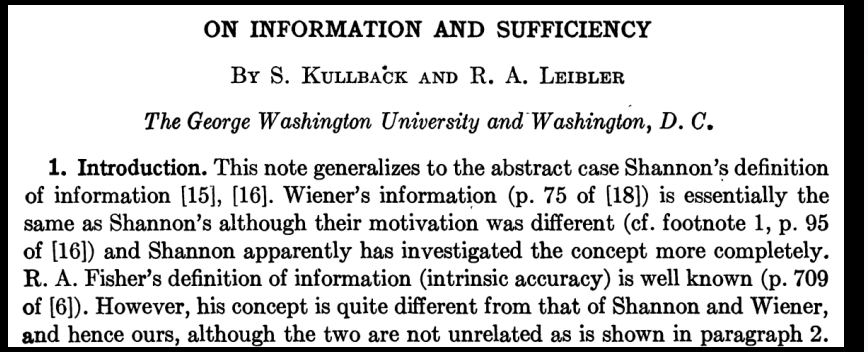
→ Stable Learning

Introduction to the inventors

In mathematical statistics, the Kullback–Leibler (KL) divergence (also called relative entropy and I-divergence[1]), denoted $D_{KL}(P||Q)$, is a type of statistical distance.

A measure of how one probability distribution P is different from a second, reference probability distribution Q.

Kullback, S.; Leibler, R.A. (1951).
"On information and sufficiency".
Annals of Mathematical Statistics. 22 (1): 79–86.
doi:10.1214/aoms/1177729694



ON INFORMATION AND SUFFICIENCY

By S. Kullback and R. A. Leibler

The George Washington University and Washington, D. C.

1. Introduction. This note generalizes to the abstract case Shannon's definition of information [15], [16]. Wiener's information (p. 75 of [18]) is essentially the same as Shannon's although their motivation was different (cf. footnote 1, p. 95 of [16]) and Shannon apparently has investigated the concept more completely. R. A. Fisher's definition of information (intrinsic accuracy) is well known (p. 709 of [6]). However, his concept is quite different from that of Shannon and Wiener, and hence ours, although the two are not unrelated as is shown in paragraph 2.

Solomon Kullback

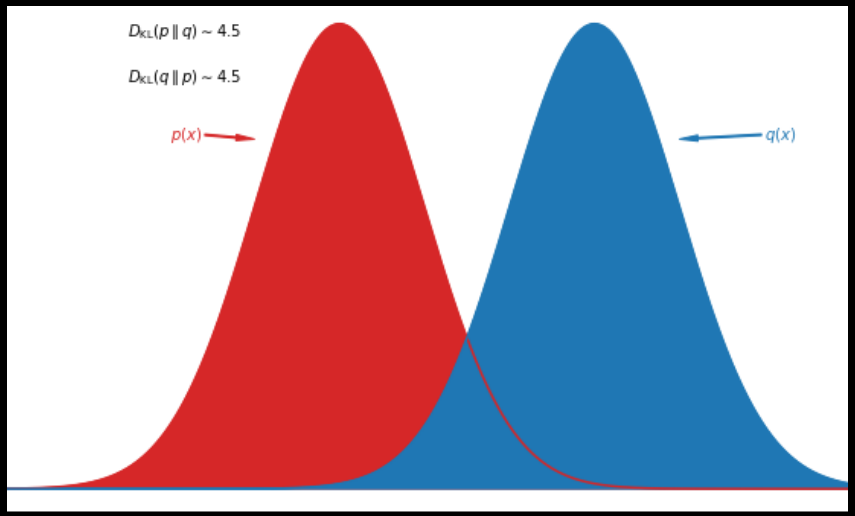
Born	April 3, 1907 Brooklyn, New York
Died	August 5, 1994 (aged 87) Boynton Beach, Florida
Citizenship	American
Alma mater	City College of New York (B.A., 1927; M.A., 1929) George Washington University (Ph.D., Mathematics, 1934)
Known for	Work in Information theory, Kullback–Leibler divergence
Scientific career	
Fields	cryptanalysis, mathematics, information theory
Institutions	George Washington University, National Security Agency

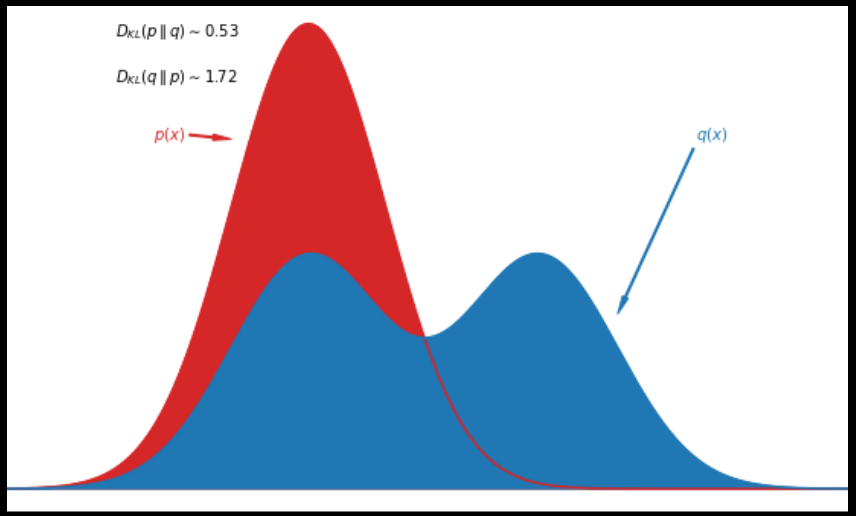
Richard Leibler

Born	March 18, 1914 Chicago, Illinois
Died	October 25, 2003 (aged 89) Reston, Virginia
Alma mater	Northwestern University (A.M., Mathematics) University of Illinois (Ph.D., Mathematics, 1939)
Known for	Kullback–Leibler divergence
Scientific career	
Fields	cryptanalysis, mathematics
Institutions	United States Navy, Princeton University, National Security Agency, Institute for Defense Analysis

Applications of KL Divergence

두 분포가 있을 경우, 그 차이를 어떻게 측정할까???





두 분포의 차이를 측정할 수 있다면?

Reinforce Learning:

Can measure the difference between two policy distributions

Deep Learning:

Can optimize the difference between $Y(P_{Label})$ and $\hat{Y}(P_{\theta})$ to be minimized

KL Divergence Definition

Definition

기준이 되는 분포

$$D(p||q) = \mathbb{E}_{p}\left[\log \frac{p(X)}{q(X)}\right] = \sum_{x \in \mathcal{X}} p(x) \times \log \frac{p(x)}{q(x)}$$

, *where* p & q *are probability distribution*

X *is random variable* and $0 \log \frac{0}{0} = 0, \quad 0 \log \frac{0}{q} = 0, \quad 0 \log \frac{p}{0} = \infty$

분자와 분모를 바꿔서 표현해도 무방합니다 ^^.

$$D(p||q) = \mathbb{E}_{p}\left[\log \frac{p(X)}{q(X)}\right] = -\sum_{x \in \mathcal{X}} p(x) \times \log \frac{q(x)}{p(x)}$$

Distance vs. Divergence

$Note$:

$In\ \ general$

$$D(p||q) \neq D(q||p)$$

거리 개념을 사용하는 것에
약간의 이견이 존재

Euclidean distance 관점에서는 틀린 말

일반적 거리 관점에서는 맞는 말

유명한 책 'Element of Information Theory' 그리고
Online Wiki 에서는 KL Divergence를
KL distance 라고 표현

In information geometry, a divergence is a kind of statistical distance: a binary function which establishes the separation from one probability distribution to another on a statistical manifold.

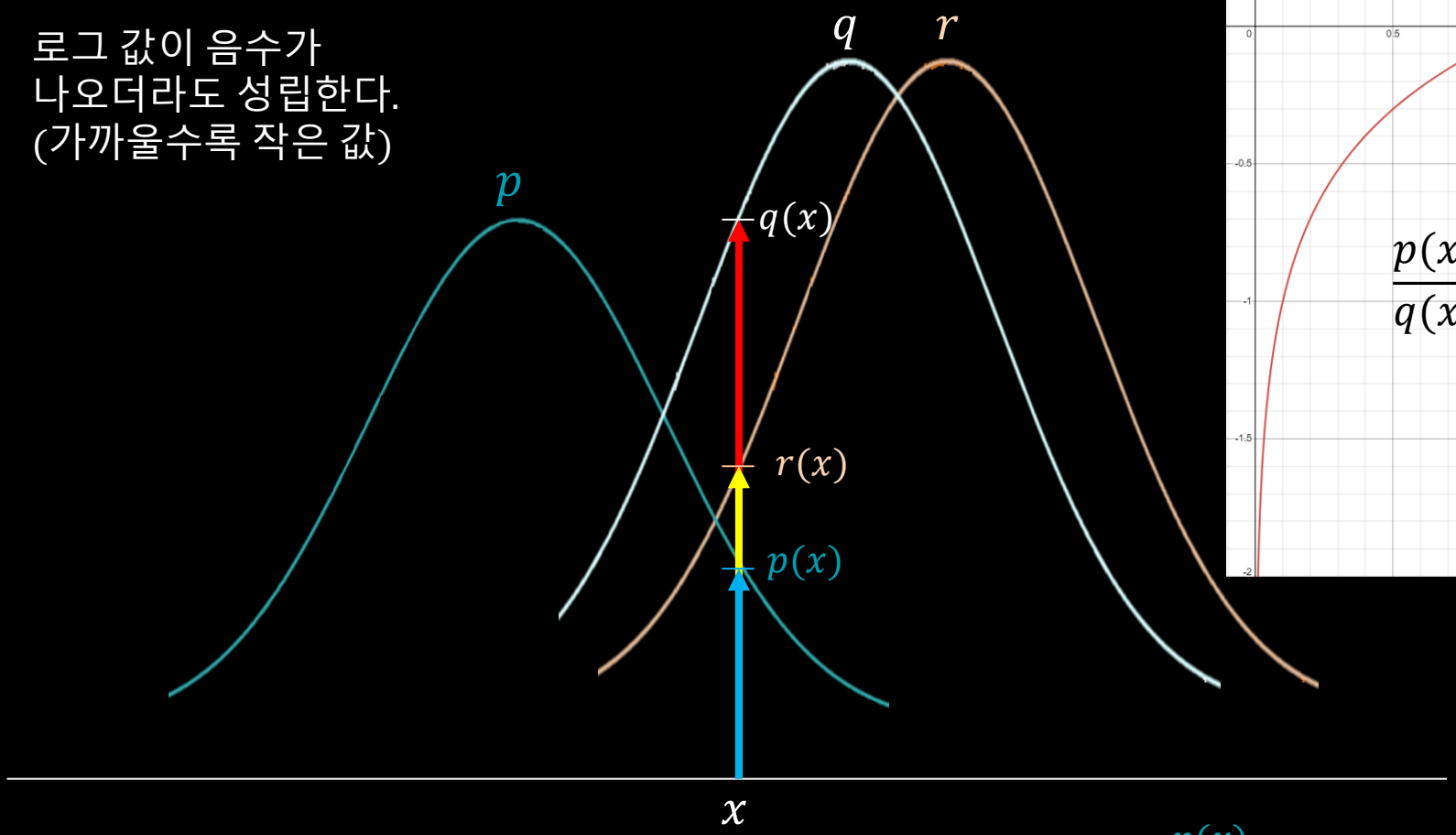
The simplest divergence is squared Euclidean distance (SED), and divergences can be viewed as generalizations of SED.

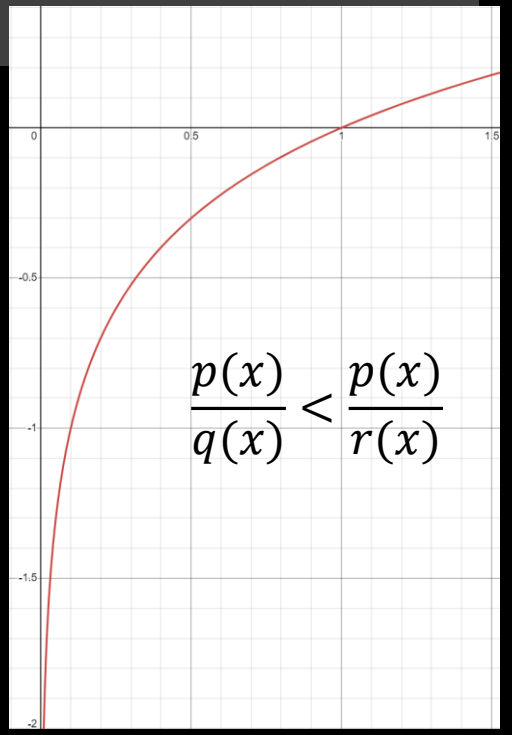
The other most important divergence is relative entropy (also called Kullback–Leibler divergence)

Source: https://en.wikipedia.org/wiki/Divergence_(statistics)

손으로 구해보는 KLD (1/2)

로그 값이 음수가
나오더라도 성립한다.
(가까울수록 작은 값)



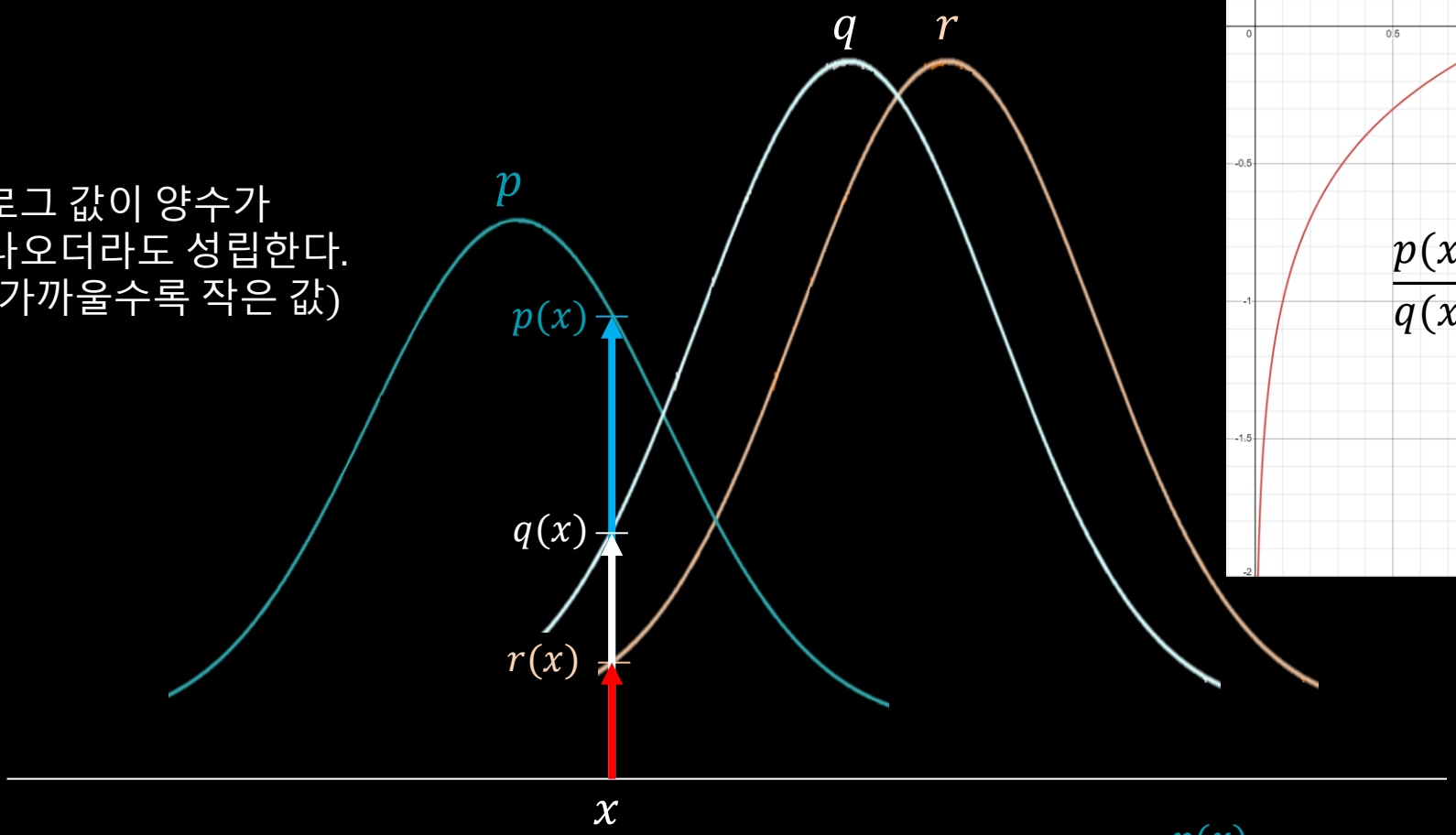


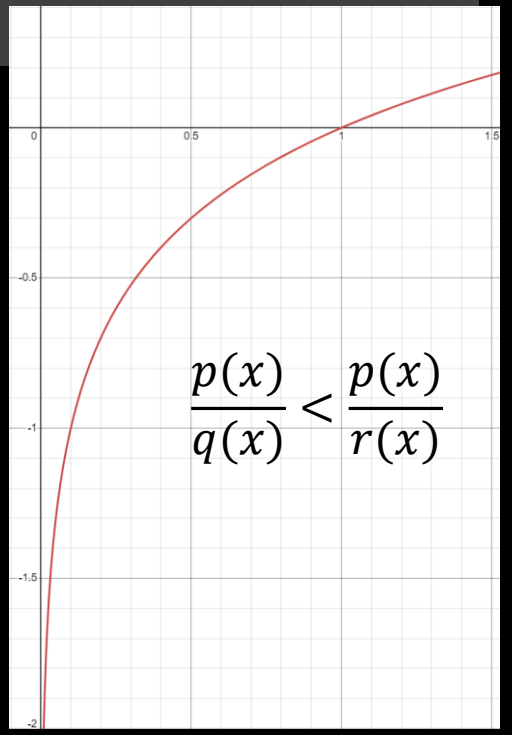
$$D(p||q) = p(x) \times \log \frac{p(x)}{q(x)}$$

$$D(p||r) = p(x) \times \log \frac{p(x)}{r(x)}$$

손으로 구해보는 KLD (2/2)

로그 값이 양수가
나오더라도 성립한다.
(가까울수록 작은 값)





$$\frac{p(x)}{q(x)} < \frac{p(x)}{r(x)}$$

$$D(p||q) = p(x) \times \log\frac{p(x)}{q(x)}$$

$$D(p||r) = p(x) \times \log\frac{p(x)}{r(x)}$$

Lower Bound of KL Divergence

Using Gibbs' Inequality

$$\log t \leq t - 1, \forall\, t > 0$$

Proof

Let $t = \frac{Q(x)}{P(x)}$

$$\log \frac{Q(x)}{P(x)} \leq \frac{Q(x)}{P(x)} - 1$$

$$P(x) \log \frac{Q(x)}{P(x)} \leq Q(x) - P(x)$$

$$\sum_x P(x) \log \frac{Q(x)}{P(x)} \leq \sum_x \big(Q(x) - P(x)\big)$$

$$\sum_x P(x) = \sum_x Q(x) = 1$$

$$\sum_x P(x) \log \frac{Q(x)}{P(x)} \leq 0$$

Therefore,

$$\sum_x P(x) \log \frac{P(x)}{Q(x)} \geq 0$$

Using Log Sum Inequality

$$\sum_{i=1}^{n} a_i \log \frac{a_i}{b_i} \geq \left(\sum_{i=1}^{n} a_i\right) \log \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}$$

Proof

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

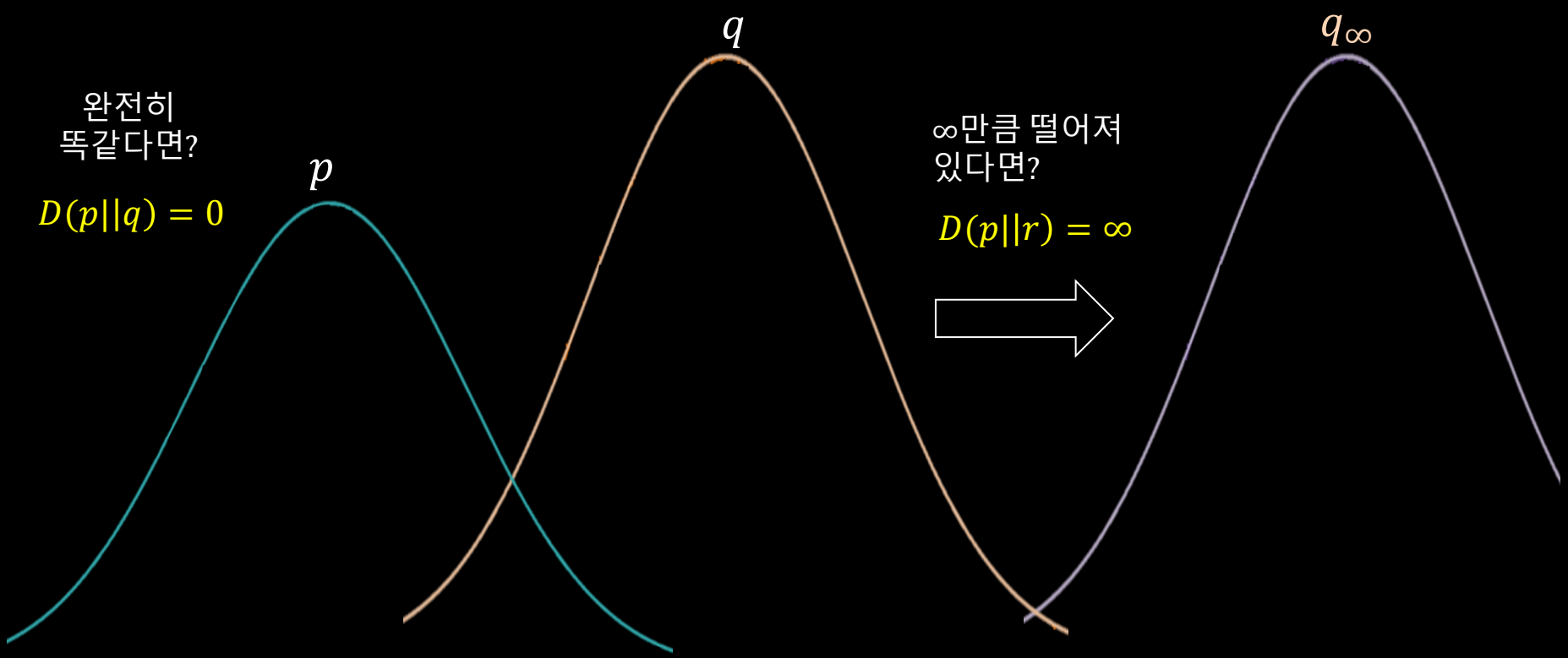
$$\geq \left(\sum_{i=1}^{n} p_i\right) \log \frac{\sum_x p(x)}{\sum_x q(x)}$$

$$= 1 \times \log \frac{1}{1} = 0$$

Therefore,

$$D(p||q) \geq 0$$

KL Divergence Visualization



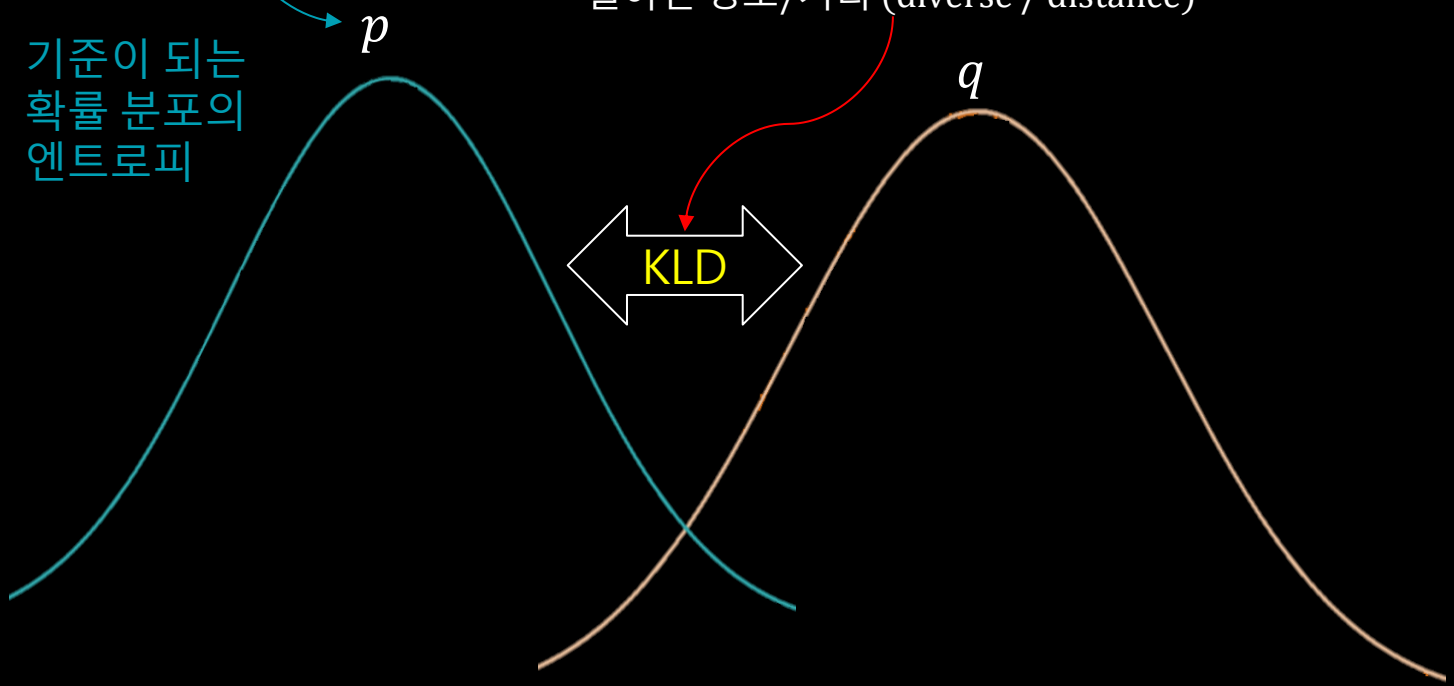
$$D(p||r) = p(x) \times \log\frac{p(x)}{q(x)}$$

KLD & Cross Entropy?

$$H(p,q) = H(p) + D(p||q)$$

$H(p)$: 기준이 되는 확률 분포의 엔트로피

$D(p||q)$: 비교 대상이 되는 확률 분포와 떨어진 정도/거리 (diverse / distance)



KL Divergence: Mathematical Interpretation

Discrete version:

$$D_{KL}(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

RL version:

$$D_{KL}(\pi_{old}||\pi_{new})$$

$$= \sum_a \pi_{old}(a|s) \log \frac{\pi_{old}(a|s)}{\pi_{new}(a|s)}$$

Trust Region Idea

$$D_{KL}(\pi_{old}||\pi_{new}) \leq \delta$$

Small KL Divergence

→ Similar Policies

→ Stable Learning

Trust Region Policy Optimization (TRPO)

Trust Region Policy Optimization (TRPO)

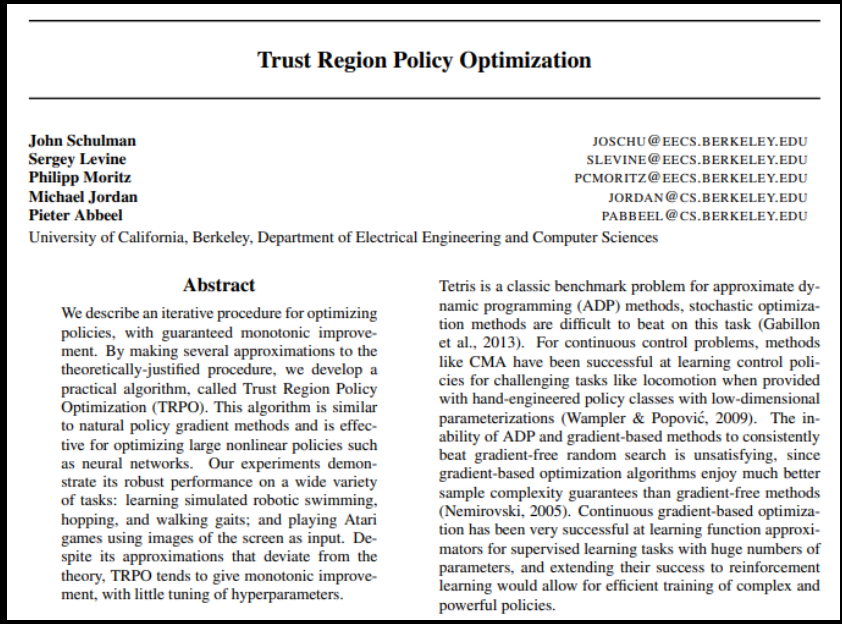
- OpenAI (2015) 공동창업자
- ChatGPT 학습과정 설계
- 엔트로픽 이직(2024)
- Thinking Machine Lab(2025)
- 현재?

⋮

John Schulman et al.

UC Berkeley, 2015, International

Conference on Machine Learning (ICML)



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

We describe an iterative procedure for optimizing policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified procedure, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is similar to natural policy gradient methods and 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.

Tetris is a classic benchmark problem for approximate dynamic 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.

Motivations:

- Large policy updates can cause unstable learning and performance collapse.
- Vanilla policy gradient methods lack a mechanism to safely constrain policy changes.

Contributions:

- Large policy updates can cause unstable learning and performance collapse.
- Vanilla policy gradient methods lack a mechanism to safely constrain policy changes.

TRPO Objective Function

Maximize policy **improvement** while keeping the new policy **close to the old policy**.

Objective

Find a new policy that **improves expected return** as much as possible.

- π_{old} : Previous (old) policy
- π_θ : New policy parameterized by θ
- $A(s,a)$: Advantage function
- D_{KL} : KL divergence
- δ : Maximum allowed divergence

TRPO Optimization Problem

Maximize (Policy Improvement)

$$\max_\theta \quad \mathbb{E}_{s\sim\pi_{old},\, a\sim\pi_{old}} \left[\frac{\pi_\theta(a|s)}{\pi_{old}(a|s)} A(s,a) \right]$$

subject to (Trust Region Constraint)

$$D_{KL}(\pi_{old} \,||\, \pi_\theta) \le \delta$$

KL divergence between old and new policy must be smaller than a threshold δ.

What Does It Mean?

Objective Term

- $\frac{\pi_\theta(a|s)}{\pi_{old}(a|s)}$: Importance ratio
- $A(s,a)$: Advantage (how good action a is in state s)
- Increase probability of **good** actions ($A>0$)
- Decrease probability of **bad** actions ($A<0$)

Constraint Term

- Limits how much the new policy can **deviate** from the old policy.
- Prevents large, risky updates and ensures **stable learning**.

Intuition

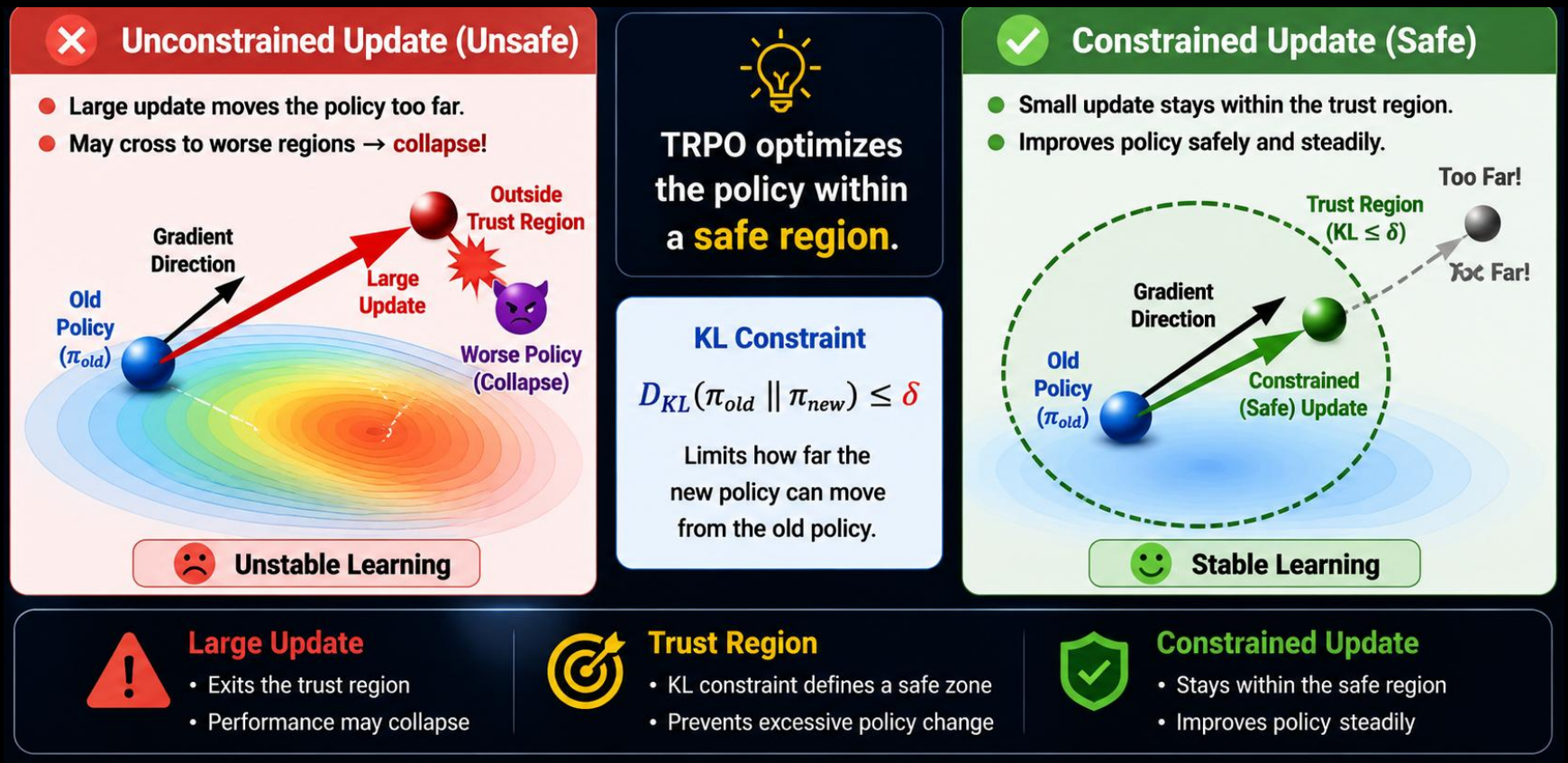
Improve the policy in the direction that **increases expected return**.

Do not change the policy too much (stay within the **trust region**).

→ **Improve the policy, but do not move too far.**

★ TRPO transforms **unstable** policy optimization into a **constrained** optimization problem

Geometry of Trust Region



Why TRPO Is Computationally Expensive?

1 Local Approximation (2nd Order)

Approximate the objective around θ_{old}

$$L(\theta) \approx L(\theta_{old}) + g^T \Delta\theta + \frac{1}{2} \Delta\theta^T H \Delta\theta$$

g : gradient

H : Hessian matrix

Second-order approximation form of Taylor expansion

$$L(\theta) \approx L(\theta_{old}) + g^T \Delta\theta + \frac{1}{2} \Delta^T H \Delta\theta$$

현재 위치에서의 objective 값 기준점(base value)

gradient 방향으로 얼마나 이동했는가? 1차 변화량 (선형 근사)

방향 $\Delta\theta$로 움직였을 때의 곡률 기반 변화량. Gradient는 증가 방향만 알려주지만 Hessian은 곡률(curvature)이 얼마나 급하게 휘는지(위험한지, KLD) 민감도 제공

g: Gradient vector,

the first derivative,

$$g = \nabla_\theta L(\theta)$$

"Where should we go?"

H: Hessian matrix,

the second derivative,

$$H = \nabla_\theta^2 L(\theta)$$

"How careful should we be?"

- $L(\theta)$: Current Objective (expected reward or surrogate objective)
- θ_{old}: Policy params before update
- θ : Current params of policy network
- $\Delta\theta$: Variation of params

Why TRPO Is Computationally Expensive?

1 Local Approximation (2nd Order)

Approximate the objective around θ_{old}

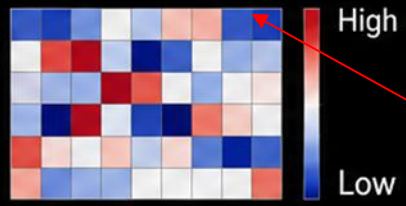
$$L(\theta) \approx L(\theta_{old}) + g^T \Delta\theta + \frac{1}{2} \Delta\theta^T H \Delta\theta$$

g : gradient

H : Hessian matrix

2 Hessian Matrix (Second Derivative)

$$H = \nabla_\theta^2 D_{KL}(\pi_{old} \| \pi_\theta)|_{\theta_{old}}$$



Captures the curvature of the constraint (KL)

KL-divergence의 곡률(curvature)을 측정해서 policy update를 안전하게 만들자!

그 방향이 얼마나 위험하게 휘어져 있는가?

$$H = \nabla_\theta^2 D_{KL}(\pi_{old} || \pi_\theta)\Big|_{\theta_{old}}$$

현재 policy 위치에서 Hessian 계산

- 현재 위치 주변의 curvature 측정
- local approximation 수행

자동차로 보는 미분의 직관

1차 미분: 속도 (얼마나 빨리 움직이나?)

2차 미분: 가속도 (속도가 얼마나 빠르게 변하나?)

- H: Hessian matrix (curvature information of KLD)
- ∇_θ^2: Second derivative (how steep the curve is)
- D_{KL}: The difference between old & new policy

$$D_{KL}(\pi_{old} || \pi_{new}) \leq \delta$$

Why TRPO Is Computationally Expensive?

1 Local Approximation (2nd Order)

Approximate the objective around θ_{old}

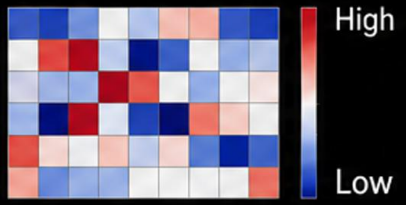
$$L(\theta) \approx L(\theta_{old}) + g^T \Delta\theta + \frac{1}{2} \Delta\theta^T H \Delta\theta$$

g : gradient

H : Hessian matrix

2 Hessian Matrix (Second Derivative)

$$H = \nabla_\theta^2 D_{KL}(\pi_{old} \| \pi_\theta)|_{\theta_{old}}$$



Captures the curvature of the constraint (KL)

3 Solve Linear System (Using Conjugate Gradient)

Find step direction $\Delta\theta$ by solving:

$$H \Delta\theta = g$$

Solved efficiently with Conjugate Gradient (CG) (no explicit matrix inverse)

4 Compute Step Size (Line Search)

Find the largest step size α that satisfies the KL constraint

$$D_{KL}(\pi_{old} \| \pi_{\theta_{old}+\alpha\Delta\theta}) \leq \delta$$

(Backtracking Line Search)

Computational Issues:

- Expensive optimization
- Complex implementation
- Difficult debugging
- Hard hyperparameter tuning

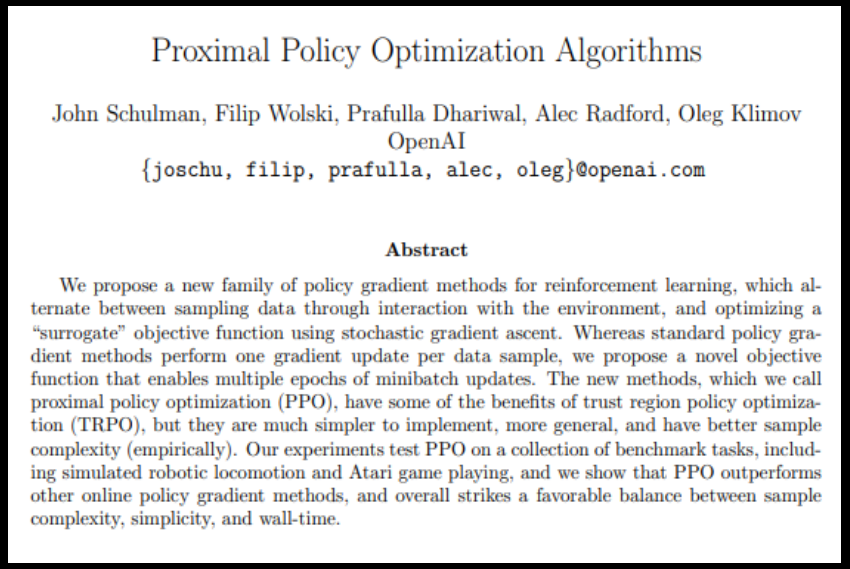
Practical Issues:

- Slow training
- Large memory usage
- Hard to scale

Can we achieve similar stability using a simpler method?

Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO)



Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI
{joschu, filip, prafulla, alec, oleg}@openai.com

Abstract

We propose a new family of policy gradient methods for reinforcement learning, which alternate between sampling data through interaction with the environment, and optimizing a "surrogate" objective function using stochastic gradient ascent. Whereas standard policy gradient methods perform one gradient update per data sample, we propose a novel objective function that enables multiple epochs of minibatch updates. The new methods, which we call proximal policy optimization (PPO), have some of the benefits of trust region policy optimization (TRPO), but they are much simpler to implement, more general, and have better sample complexity (empirically). Our experiments test PPO on a collection of benchmark tasks, including simulated robotic locomotion and Atari game playing, and we show that PPO outperforms other online policy gradient methods, and overall strikes a favorable balance between sample complexity, simplicity, and wall-time.

John Schulman et al.

(Same to TRPO first author)

OpenAI, 2017, arXiv preprint,

No Conference, No Journal, But Widely adopted in industry and research

Motivations:

- TRPO achieves stable learning but requires expensive second-order optimization.
- More practical policy optimization algorithm was needed for large-scale deep RL.

Contributions:

- Introduced clipped policy objective for stable first-order policy optimization.
- Achieved TRPO-like stability with significantly simpler implementation and lower cost.

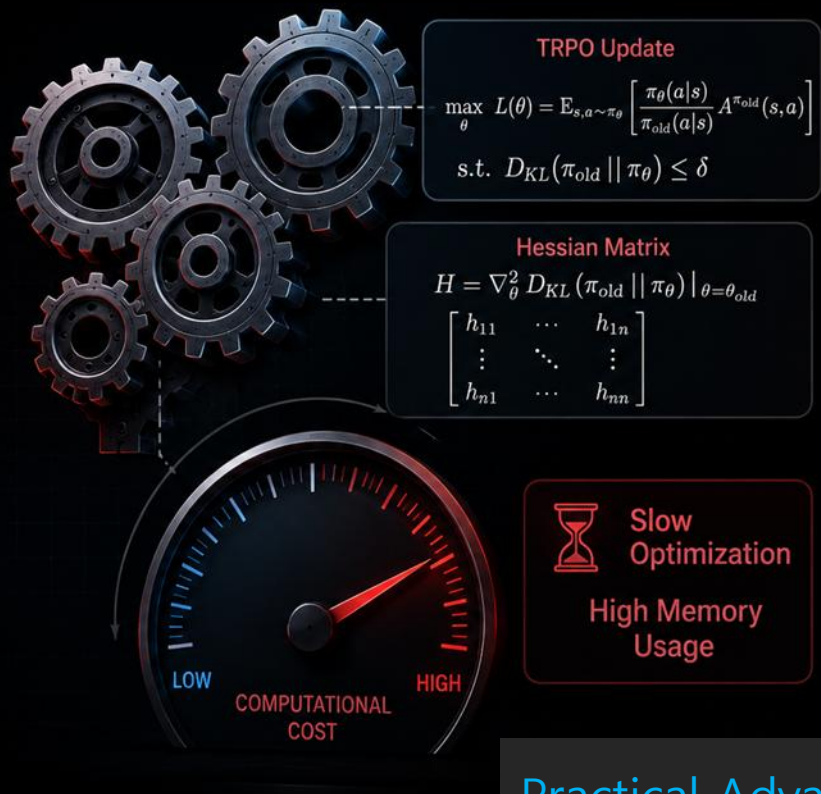
PPO Motivation

Second-order optimization
Requires expensive curvature information.

Hessian approximation
Computing or approximating the Hessian is costly.

Complex implementation
Involves conjugate gradient, line search, and KL constraints.

High computational cost
Slow optimization and high memory usage.



Computational Issues:

- **PPO simplifies TRPO**
- Safe update region while
- Simple gradient ascent

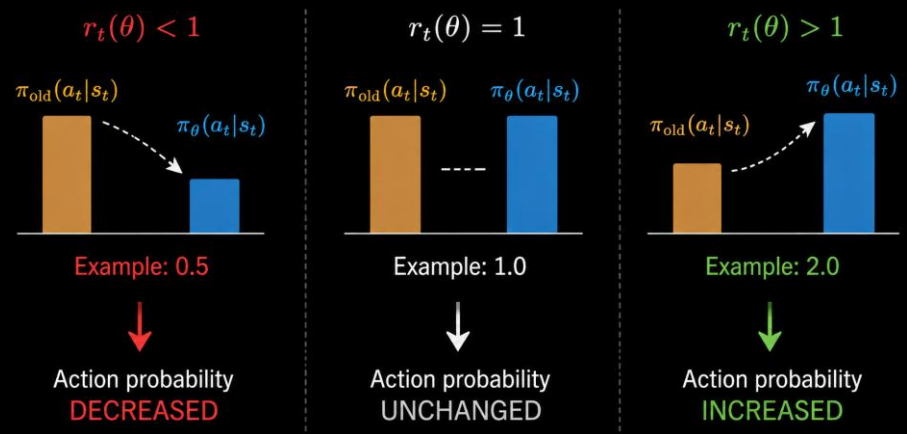
Practical Advantages:

- First-order optimization
- Easy implementation
- Stable learning
- Approximate trust region

Stable Policy Updates with Clipping

Policy Ratio

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}$$

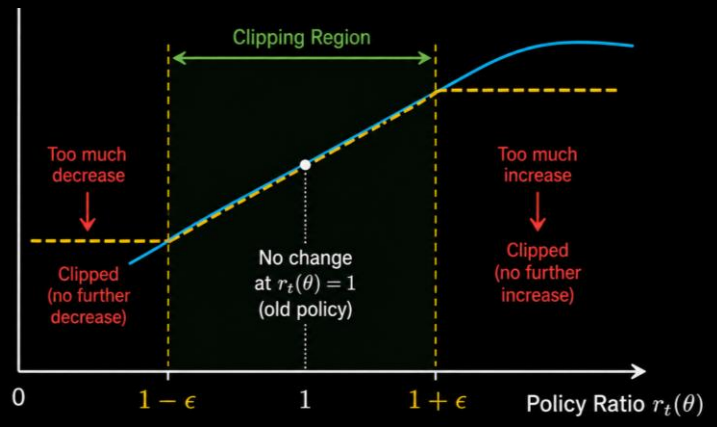


Measures how much the new policy changes

- Ratio > 1 → action probability increased
- Ratio < 1 → action probability decreased

Clipping Trick

$$clip(r_t(\theta), 1-\epsilon, 1+\epsilon)$$



Prevent excessively large policy updates

PPO Clipped Objective Function

PPO stabilizes policy learning using a simple clipping mechanism.

Original Objective

$$r_t(\theta) A_t$$

- Increase probability of good actions
- Decrease probability of bad actions

Intuition

- $A_t > 0$ → Increase $\pi_\theta(a_t|s_t)$ Good action
- $A_t < 0$ → Decrease $\pi_\theta(a_t|s_t)$ Bad action

PPO Clipped Objective

$$L^{CLIP}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)A_t,\ clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right]$$

$r_t(\theta)A_t$ (Unclipped) , $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t$ (Clipped)

$\min(\ ,\ )$

Intuition

- Take the minimum of the unclipped and clipped objectives.
- Prevents the objective from increasing when the update is too large.
- Enables stable and safe policy updates.

If update is small

- behave like normal policy gradient

If update becomes too large

- clipping activates
- objective stops increasing

Results:

- table learning
- safer optimization

Understanding the Min Operator

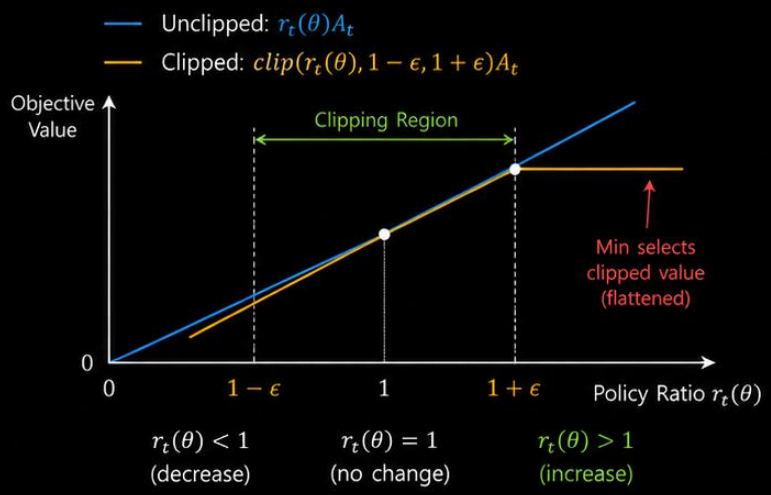
1) Positive Advantage ($A_t > 0$)

Good action

Goal: Increase action probability ($r_t(\theta) > 1$)

Problem: Can increase too much (unstable updates)

$$L^{CLIP}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)A_t,\ clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right]$$



Key point ($A_t > 0$):
PPO prevents over-optimization.
Even if $r_t(\theta)$ is very large, the objective is capped so updates stay within a safe range.

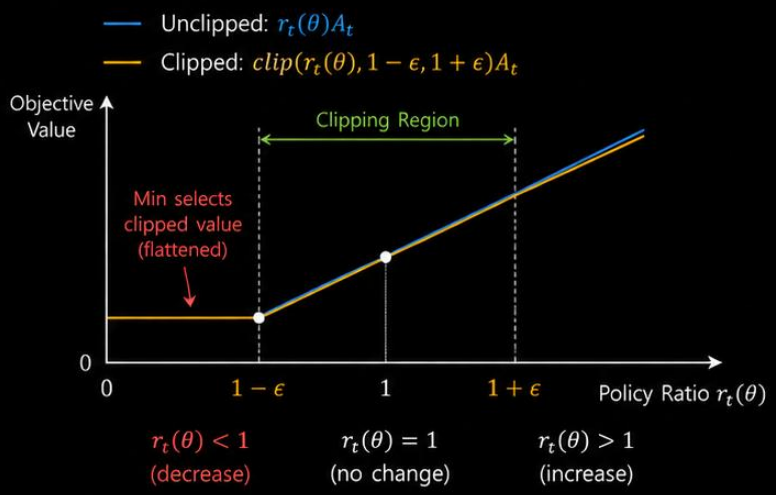
2) Negative Advantage ($A_t < 0$)

Bad action

Goal: Decrease action probability ($r_t(\theta) < 1$)

Problem: Can decrease too much (unstable updates)

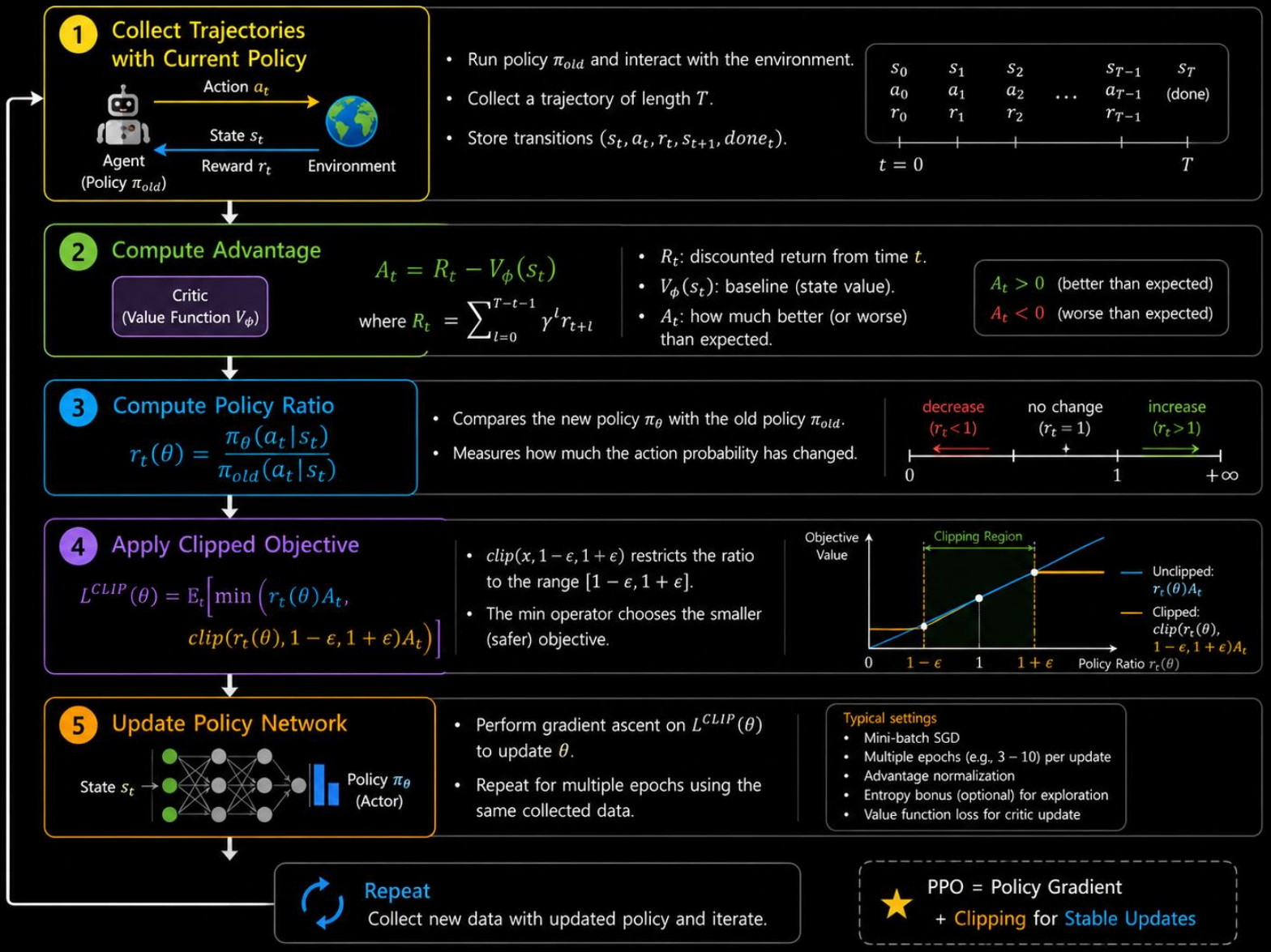
$$L^{CLIP}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)A_t,\ clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right]$$



Key point ($A_t < 0$):
PPO prevents over-penalization.
Even if $r_t(\theta)$ is very small, the objective is bounded so updates do not become destructive.

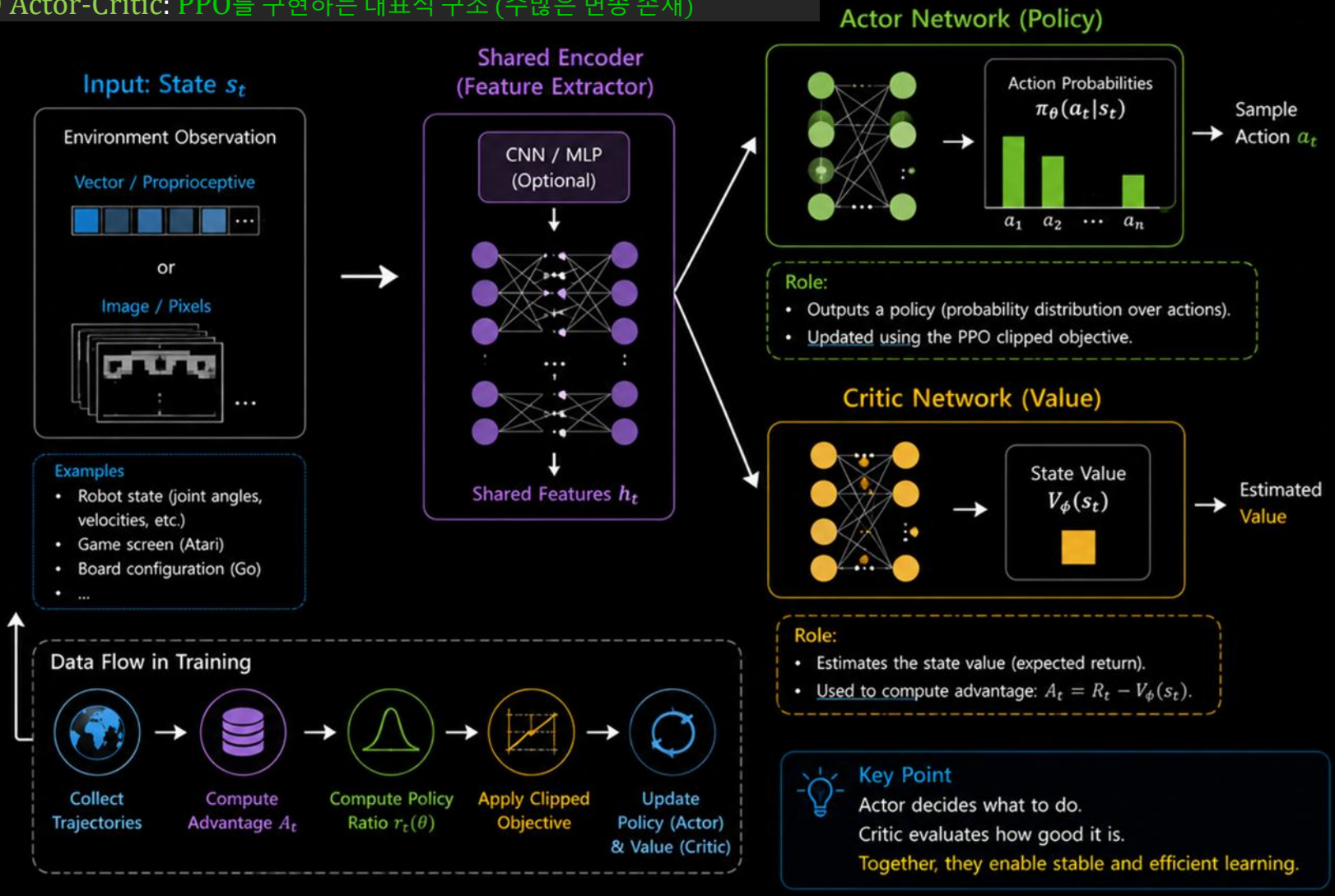
Overall Intuition:
The min operator makes PPO pessimistic.
It chooses the smaller (safer) objective to avoid overly large updates in both directions, leading to more stable and reliable policy learning.

PPO Training Pipeline

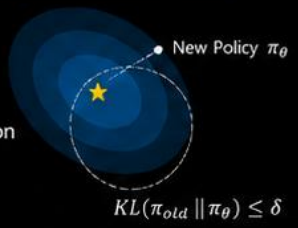
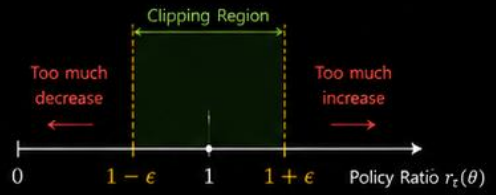


PPO Actor–Critic Architecture

- PPO: Reinforce Learning Algorithm (너무 급하게 바꾸지 않는다는 철학)
- PPO Actor-Critic: PPO를 구현하는 대표적 구조 (수많은 변종 존재)



PPO vs TRPO Comparison

	TRPO (Trust Region Policy Optimization)	PPO (Proximal Policy Optimization)
Main Idea	Constrained optimization: maximize the objective while keeping the policy within a trust region measured by KL divergence. 	Unconstrained optimization: use clipping to limit how much the new policy can improve (or worsen) the objective. 
Objective	maximize $L(\theta) = \mathbb{E}\left[r_t(\theta)A_t\right]$	maximize $L^{CLIP}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)A_t,\ clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right]$
Constraint	KL divergence constraint $KL(\pi_{old} \| \pi_\theta) \leq \delta$	No explicit constraint (clipping implicitly limits large updates)
How It Works	• Solve a constrained optimization problem • Ensure the new policy stays within the trust region	• Use clipped objective • Prevent large policy updates via the clip function
Optimization Method	Conjugate Gradient to solve $\max_\theta \ L(\theta) \ \ s.t. \ \ KL \leq \delta$	Standard first-order methods (SGD / Adam)
Computational Cost	High (requires Fisher information matrix and iterative solver)	Low (no need for Fisher matrix or second-order solver)
Implementation Complexity	High (more code, more hyperparameters)	Low (simple and easy to implement)
Sample Efficiency	High (due to accurate trust region constraint)	Good (slightly lower than TRPO in theory, but often comparable in practice)
Practical Use	Less commonly used (complex to implement and tune)	Most widely used in practice (stable, simple, and scalable)

Summary

Goal
Both aim to improve the policy while avoiding too large updates.

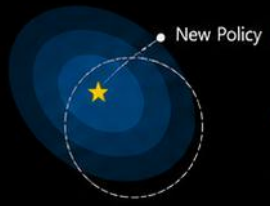
Difference
TRPO enforces a hard constraint (theoretical). PPO uses a soft constraint (practical).

Takeaway
PPO keeps the spirit of TRPO but is much simpler and more practical, which is why it became the default choice in modern RL.

Why PPO Became Popular

TRPO (2015)

Theoretically principled

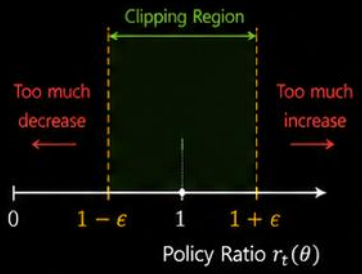


$KL(\pi_{old} \,||\, \pi_\theta) \leq \delta$

- ✓ Strong theoretical guarantees (KL constraint)
- ✓ Stable and monotonic improvement
- ✗ Complex to implement
- ✗ Requires Fisher information matrix
- ✗ High computational cost

PPO (2017)

Practical and effective



- ✓ Simple clipping trick (no explicit constraint)
- ✓ Easy to implement
- ✓ Low computational cost
- ✓ Sample efficient
- ✓ Performs close to TRPO in practice

Impact

Widespread adoption

- State-of-the-art results on a wide range of tasks
- Default algorithm in many RL libraries and baselines
- Strong empirical performance with minimal tuning effort
- Scalable to high-dimensional and continuous control tasks

Key Reasons PPO Became the De-facto Algorithm

1 Simplicity

Easier to implement than TRPO.

No need to compute Fisher matrix or solve constrained optimization.

2 Efficiency

Lower computational cost and memory usage.

Faster training in practice.

3 Sample Efficiency

Uses large mini-batches and multiple epochs on the same data.

Better data utilization.

4 Strong Performance

Achieves performance comparable to TRPO across many benchmarks (Ant, Humanoid, MuJoCo, Atari, etc.).

5 Robustness

Clipping prevents excessively large or harmful updates.

More stable learning in practice.

6 Ease of Use (Tuning)

Few hyperparameters are sensitive.

Works well across diverse environments with minimal tuning.

Takeaway

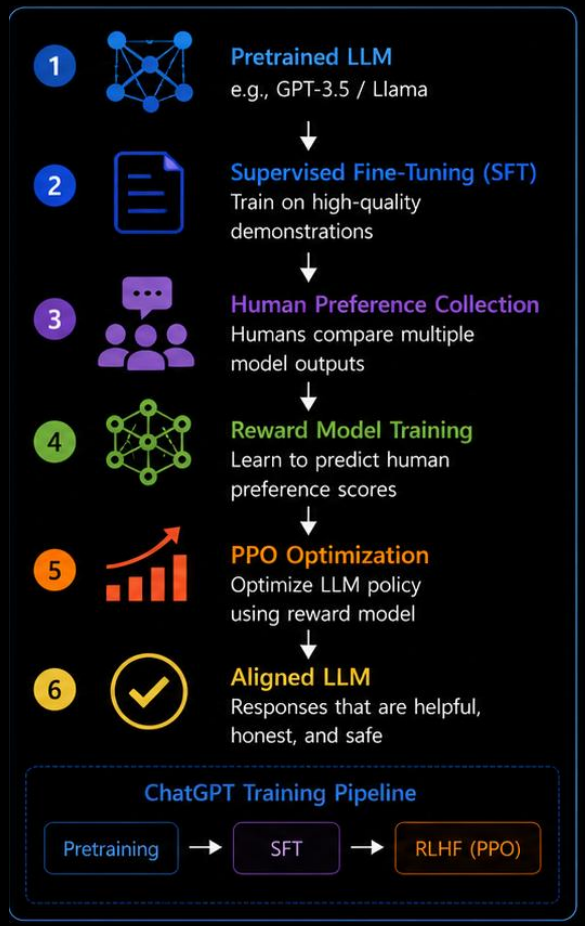
PPO keeps the theoretical spirit of TRPO while removing implementation barriers.
Its simplicity, efficiency, and strong empirical performance made it the most widely used on-policy RL algorithm in both research and industry.

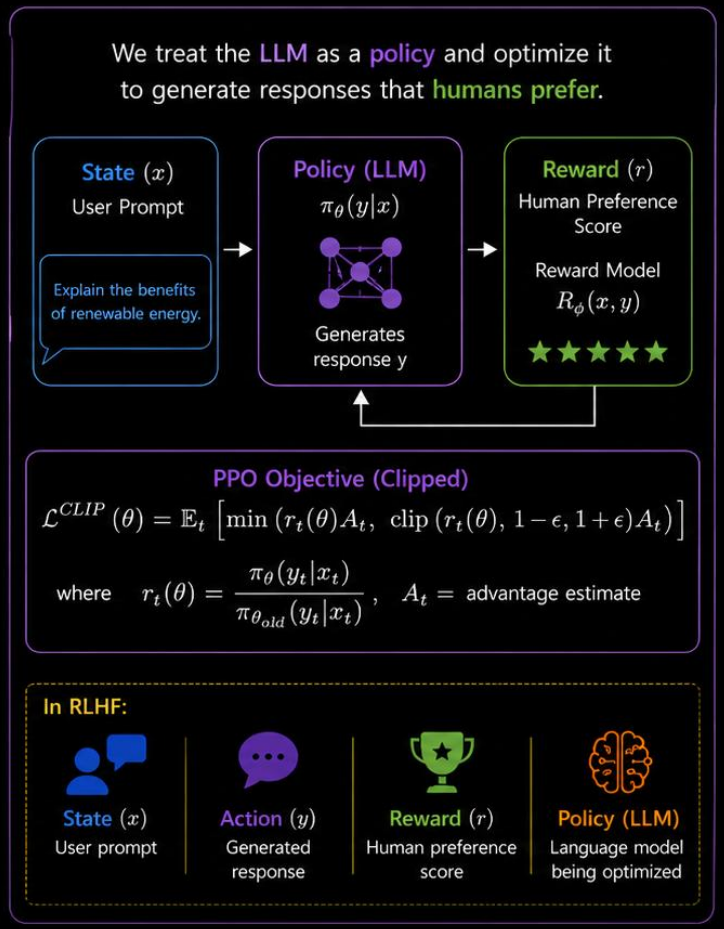
PPO

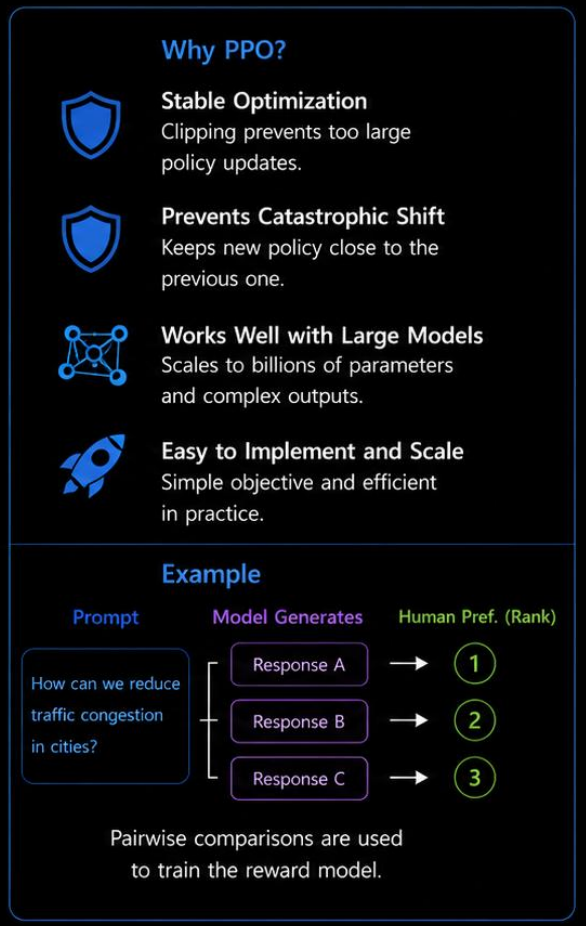
RLHF for Large Language Models

PPO is not only used in robotics or games.

It is also used to align large language models with human preferences.







Challenges of RLHF

1 Reward Hacking

The model learns to maximize the reward instead of fulfilling the true intent.

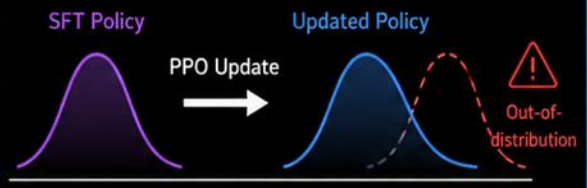
Examples

- ⊗ Finds loopholes in the reward model (e.g., verbose but unhelpful responses)
- ⊗ Exploits flaws or biases in human feedback
- ⊗ Looks good to the reward model, but not actually helpful to users

Optimizing reward
≠ Optimizing human intention

2 Distribution Shift & Instability

Policy updates can move the model to unseen regions, causing instability.



Issues

- ⊗ Performance drops in new regions
- ⊗ Catastrophic drift with large updates
- ⊗ Sensitivity to hyperparameters (e.g., learning rate, clipping range)

Need stable optimization
to prevent degradation.

3 Hallucination & Alignment

The model may generate plausible but incorrect or misleading outputs.

Hallucination

Q: Who won the Nobel Prize in Physics in 2023?

A: Jane Doe. (Incorrect)

Alignment Issues

Q: Tell me how to make harmful substances.

A: Sure, here is how ... (Unsafe)

Issues

- ⊗ Reward model may not capture all aspects of helpfulness and safety
- ⊗ Hard to detect subtle or long-term harms
- ⊗ Generalization to unseen prompts is difficult

Alignment is an open problem,
not fully solved by current RLHF.

Key Takeaway

RLHF is powerful, but optimizing for human feedback is an imperfect proxy for true human values.
Building more robust, generalizable, and safe alignment methods remains a major research frontier.

PPO Limitations & Future Research

PPO Limitations:

- On-policy inefficiency
- Requires many environment interactions
- Hyperparameter sensitivity
- Exploration difficulty

Research Topics:

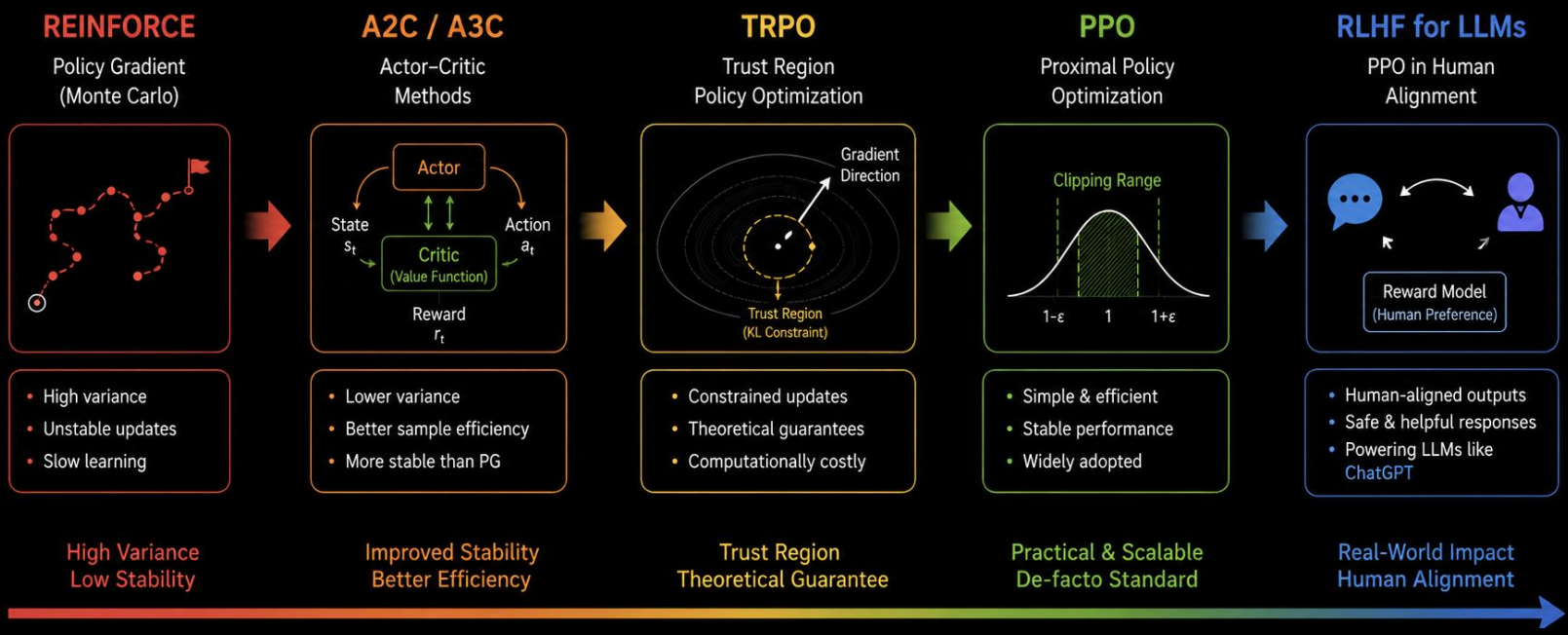
- Reduce environment interaction cost
- Improve data reuse efficiency
- Curiosity-driven learning
- Learning without online environment interaction
- Training from static datasets
- Cooperation and competition between agents
- Scalable decentralized learning

⋮

Summary

Key Insights:

- Large policy updates can destabilize learning
- Trust-region methods improve stability
- PPO simplifies TRPO using clipping
- PPO became the de-facto on-policy RL algorithm
- RLHF extended PPO into the LLM era



PPO bridged classical RL and modern LLM alignment.

수고하셨습니다 ..^^..