Reinforcement Learning with Constraints: From Lagrangian Relaxation to State Augmentation

MS&E 318 Presentation

Angikar Ghosal

January 27, 2026

Table of Contents

TRPO Quiz (Part 1/2)

1. What is the fundamental theoretical guarantee provided by the surrogate objective function maximized in TRPO?

A. It guarantees that the policy will explore every state in the environment.
B. It guarantees that the value function will be perfectly estimated.
C. It guarantees monotonic improvement in the expected return.
D. It guarantees that the training process will be computationally cheaper than standard Policy Gradient.

2. TRPO constrains the update step to ensure the new policy does not deviate too far from the old policy. Which metric is used to define this "Trust Region"?

A. The Euclidean distance (L_2 norm) between the parameter vectors θ_{new} and θ_{old}.
B. The Kullback-Leibler (KL) Divergence between the old and new policy distributions.
C. The difference in total expected reward between the two policies.
D. The element-wise difference in action probabilities.

TRPO Quiz (Part 2/2)

3. In the practical implementation of TRPO, computing the inverse of the Fisher Information Matrix (FIM) is computationally expensive. How does TRPO solve this?

A. It uses the Conjugate Gradient algorithm to approximate the matrix-vector product.

B. It ignores the Fisher Information Matrix and uses standard Gradient Descent instead.

C. It uses Singular Value Decomposition (SVD) to invert the matrix exactly.

D. It replaces the Fisher Information Matrix with a matrix with only non-zero elements on the diagonal.

4. How does TRPO differ from the "Natural Policy Gradient" algorithm described in the paper?

A. Natural Policy Gradient uses a fixed step size, whereas TRPO finds the step size by enforcing a constraint on the KL divergence.

B. Natural Policy Gradient maximizes the original reward term in the optimization problem, while TRPO maximizes the safety reward term in the conjugate formulation.

C. Natural Policy Gradient uses L_1 regularization to encourage sparsity of the solution, while TRPO uses L_2 to overall minimize the norm.

D. There is no difference; the authors prove that under quite weak conditions, Natural Policy Gradient is a special case of TRPO.

CPO Quiz

1. What distinguishes constrained MDPs from usual MDPs (in plain language or math equations)?
2. Write down one advantage of CPO algorithm.

Why Safe Reinforcement Learning?

The Promise of RL:

- Superhuman performance in Go, Dota 2, Starcraft.
- Solving complex control tasks from scratch.

The Reality Gap:

- Standard RL assumes "trial and error" is cheap.
- In robotics, industrial control, or healthcare, errors are **costly** or **catastrophic**.

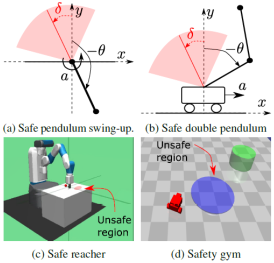

Figure 1. Panels a and b: safe pendulum environments. In both cases, θ - is the angle from the upright position, a is the action, δ - is the unsafe pendulum angle, the safety cost is the distance toward the unsafe pendulum angle, which is incurred only in the red area. Panel e: safe reacher: robot needs to avoid the unsafe region (in red). Panel d: a safety gym environment: robot needs to reach the goal (in green) while avoiding the unsafe region (in blue).

The Core Question

How do we allow an agent to maximize utility while guaranteeing (with high probability) that it remains within a defined valid region of the state space?

Mathematical Formulation: CMDPs

Standard MDP: $\mathcal{M} = \langle S, A, P, R, \gamma \rangle$.
Constrained MDP (CMDP): $\mathcal{M}_C = \langle S, A, P, R, C, \gamma, d \rangle$.

- $C(s, a)$: A cost signal (distinct from reward).
- d: The safety threshold (budget).

The Optimization Problem

$$\begin{aligned} \max_{\pi \in \Pi} \quad & J_R(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] \\ \text{s.t.} \quad & J_C(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t)\right] \leq d \end{aligned}$$

Key Difficulty: The set of feasible policies $\Pi_C = \{\pi : J_C(\pi) \leq d\}$ is generally non-convex, even if the reward function is convex.

Why TRPO? The Precursor to Safety

Before addressing explicit constraints ($J_C \leq d$), we must ensure the policy update itself is **stable**.

The Problem with Vanilla Policy Gradient

Standard Policy Gradient (e.g., REINFORCE): $\theta_{new} = \theta_{old} + \alpha \nabla J(\theta)$.

- Choosing step size α is difficult.
- Too small: Slow convergence.
- Too large: Policy "collapses" into a deterministic, suboptimal, or dangerous region.
- In Safe RL, a policy collapse usually implies catastrophic constraint violation.

TRPO Solution: Enforce a constraint on the magnitude of the policy change in distribution space, not parameter space.

Theoretical Foundation: Monotonic Improvement

Kakade & Langford (2002) proved the following identity for the expected return of a new policy $\tilde{\pi}$ vs old policy π:

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s\sim\rho_{\tilde{\pi}}, a\sim\tilde{\pi}}[A_\pi(s,a)]$$

This is hard to optimize because $s \sim \rho_{\tilde{\pi}}$ (the state distribution of the *new* policy) is unknown. TRPO uses a local approximation $L_\pi(\tilde{\pi})$:

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

The Trust Region: To trust this approximation, $\tilde{\pi}$ must be close to π. TRPO enforces:

$$\bar{D}_{\mathsf{KL}}(\pi||\tilde{\pi}) \leq \delta$$

The Practical TRPO Algorithm

We solve the following constrained optimization at each step:

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

Taylor Expansion Approximation

1. Linearize the objective: $g^T(\theta - \theta_{old})$.
2. Quadraticize the constraint: $\frac{1}{2}(\theta - \theta_{old})^T H(\theta - \theta_{old}) \leq \delta$.
3. H is the **Fisher Information Matrix (FIM)**.

Analytic Solution

$$\Delta\theta = \sqrt{\frac{2\delta}{g^T H^{-1} g}} H^{-1} g$$

TRPO Results

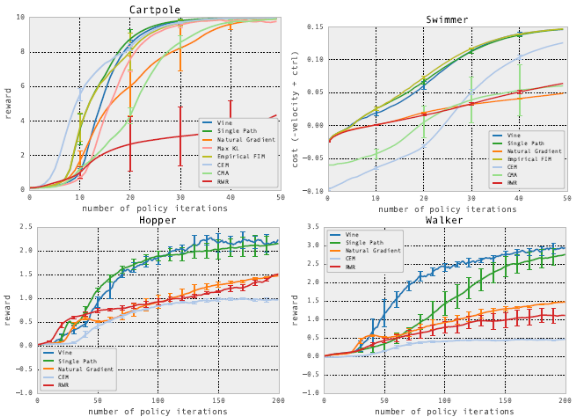

Figure 4. Learning curves for locomotion tasks, averaged across five runs of each algorithm with random initializations. Note that for the hopper and walker, a score of -1 is achievable without any forward velocity, indicating a policy that simply learned balanced standing, but not walking.

Figure: Learning curves for locomotion tasks. Note the monotonic improvement characteristic of Trust Region

Implementing TRPO Efficiently

Challenge: The Fisher Matrix H is size $|\theta| \times |\theta|$ (millions of parameters). Computing H^{-1} is $O(N^3)$.

Solution: Conjugate Gradient (CG)

- We only need to compute $x = H^{-1}g$, i.e., solve $Hx = g$.
- CG solves linear systems using only matrix-vector products (Hv).
- We can compute Hv directly without forming H explicitly (using the Fisher-vector product trick via backprop).

Connection to Safety

- TRPO creates a "Trust Region."
- **CPO (Constrained Policy Optimization)** adds a *second* linear constraint for safety: $c^T(\theta - \theta_{old}) \leq d - J_C$.
- CPO is the baseline for almost all papers discussed today.

RCPO: Reward Constrained Policy Optimization

Tessler et al., 2019

Moving from complex trust regions to simple penalties.

The Lagrangian

$$L(\theta, \lambda) = J_R(\pi_\theta) - \lambda \cdot (J_C(\pi_\theta) - d)$$

Two-Timescale Update Rule

1. **Critic (Reward & Cost):** Estimate $V_R(s)$ and $V_C(s)$.
2. **Actor (Policy θ):** Update to maximize L treating λ as fixed.
3. **Multiplier (λ):** Update using gradient ascent (Dual Ascent).

$$\lambda_{k+1} = \max(0, \lambda_k + \eta(J_C(\pi_\theta) - d))$$

Intuition: If constraints are violated, λ grows, effectively reducing the reward for unsafe actions.

RCPO Analysis: Pros and Cons

Advantages:

- **Reward Agnostic:** Invariant to reward scaling.
- **General:** Works with any Policy Gradient alg (PPO, TRPO, A3C).
- **Asymptotic Convergence:** Given enough time, it converges to a feasible policy.

Disadvantages:

- **Oscillation:** The interplay between θ and λ often leads to cycling (safe/low reward $\leftrightarrow$ unsafe/high reward).
- **Training Safety:** No guarantee that constraints are satisfied *during* training.
- **Sensitivity:** Performance highly dependent on λ's learning rate.

RCPO Results

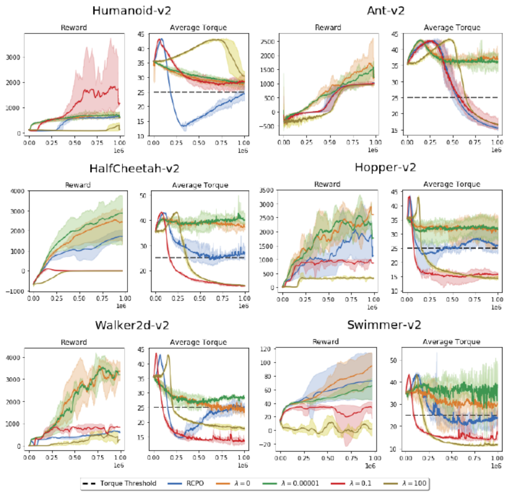

Figure 3: Mujoco with torque constraints. The dashed line represents the maximal allowed value. Results are considered valid only if they are at or below the threshold. RCPO is our approach, whereas each λ value is a PPO simulation with a fixed penalty coefficient. Y axis is the average reward and the X axis represents the number of samples (steps).

Figure: Visualizing the oscillation problem in Lagrangian methods.

Lyapunov-based Safe RL: Methodology

Chow et al., 2018

The Objective: Guarantee safety at *every* iteration, not just asymptotically.

Lyapunov Function $L_\pi(s)$

A scalar function measuring the "energy" or "risk" of the system.

$$L_\pi(x) \geq \mathcal{T}_{\pi,c}[L](x)$$

where $\mathcal{T}$ is the Bellman operator for the cost.

From Global to Local

The paper proves that ensuring the Lyapunov drift is negative:

$$\mathbb{E}[L(s_{t+1})|s_t] - L(s_t) \leq -\epsilon$$

is sufficient to guarantee the global constraint $J_C(\pi) \leq d$.

Lyapunov Approaches: Implementation & Analysis

Safe Policy Iteration (SPI)

Instead of optimizing policy blindly, project the policy update onto the set of policies that satisfy the Lyapunov decrease condition.

Implementation (Safe DQN):

- Construct a set of "safe" actions at each state based on the Lyapunov value.
- Restrict the ϵ-greedy exploration to only these safe actions.

Trade-off: Strongest theoretical safety, BUT constructing the Lyapunov function often requires solving a Linear Program (LP) or estimating complex auxiliary value functions.

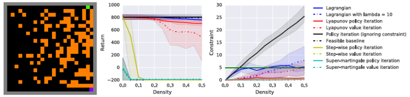

Figure 1: Results of various planning algorithms on the grid-world environment with obstacles, with x-axis showing the obstacle density. From the leftmost column, the first figure illustrates the 2D planning domain example ($\rho = 0.25$). The second and the third figures show the average return and the average cumulative constraint cost of the CMDP methods, respectively. The fourth figure displays all the methods used in the experiment. The shaded regions indicate the 80% confidence intervals. Clearly the safe DP algorithms compute policies that are safe and have good performance.

Figure: Safe exploration using Lyapunov constraints.

IPO: Interior-Point Policy Optimization

Liu et al., 2020

Philosophy: Primal-Dual methods (Lagrangians) are unstable because they solve a min-max game. Let's solve a pure maximization problem using barriers.

Logarithmic Barrier

$$\max_{\theta} \quad J_R(\pi_\theta) + \frac{1}{t}\ln(d - J_C(\pi_\theta))$$

- If $J_C(\pi) \ll d$, the barrier is small (negligible effect).
- If $J_C(\pi) \to d$, $\ln(0) \to -\infty$. The gradients explode, pushing the policy violently back into the safe set.

Results: IPO is empirically more stable than Lagrangian methods. It is less sensitive to hyperparameters (mostly just the barrier intensity t).

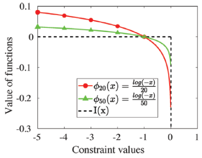

Figure 1: Value of indicator function $I(x)$ and logarithmic barrier functions $\phi(x) = \frac{-log(-x)}{t}$. The dashed line is the indicator function and two solid lines are logarithmic barrier function with $t = 20$ and $t = 50$. We get better approximation with higher t comparing these two solid lines.

CRPO: Methodology

Constraint-Rectified Policy Optimization [Xu et al., 2021]

Core Innovation

A simplified "Switching" primal approach.

The Algorithm

At iteration k, estimate current constraint violation $J_C(\pi_k)$.

1. **Case 1 (Safe):** If $J_C(\pi_k) \leq d$:

$$\pi_{k+1} \leftarrow \text{Update}(\pi_k, \nabla J_R)$$

   (Pure Reward Maximization)

2. **Case 2 (Unsafe):** If $J_C(\pi_k) > d$:

$$\pi_{k+1} \leftarrow \text{Update}(\pi_k, -\nabla J_C)$$

   (Pure Cost Minimization)

CRPO: Theoretical Guarantees & Results

Theoretical Guarantee

They prove that this randomized switching strategy achieves global convergence with an $O(1/\sqrt{T})$ rate. This is surprisingly fast for such a simple heuristic.

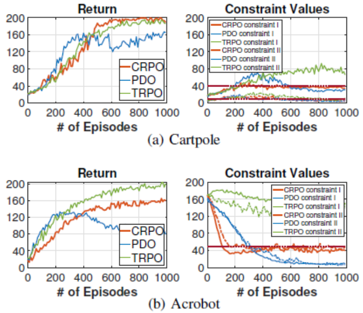

Figure 1. Average performance for CRPO, PDO, and unconstrained TRPO over 10 seeds. The red dot lines in (a) and (b) represent the limits of the constraints.

Figure: CRPO performance compared to Primal-Dual methods. Note the stability and speed.

WCSAC: Motivation

Worst-Case Soft Actor Critic [Yang et al., 2021]

The Flaw of Expectations

Critique of Previous Methods: RCPO, CPO, IPO all constrain $\mathbb{E}[\text{Cost}] \leq d$.

- Example: A drone crashes 1% of the time and flies perfectly 99% of the time.
- Expected cost is low.
- Real-world utility is zero (the drone is broken).

Conditional Value at Risk (CVaR)

Focus on the tail of the distribution (the worst $\alpha\%$ of outcomes).

$$\text{CVaR}_\alpha(C) = \mathbb{E}[C \mid C \geq \text{VaR}_\alpha(C)]$$

WCSAC: Algorithm & Results

Algorithm Overview

- Based on Soft Actor-Critic (SAC).
- Learns a **distributional critic** (Gaussians) for safety to estimate tail risk.
- Safety weights adapt based on the estimated risk.

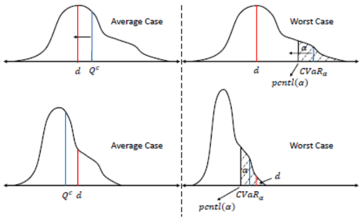

Figure 2: In the average case (left panel), the policies are optimized to ensure Q^c (blue line) is moved to the left side of the fixed boundary d (red line). In the worst case (right panel), we optimize policies to reshape the distribution of long-term costs to ensure the $CVaR_\alpha$ (blue line) measure on the left side of the fixed boundary d (red line).

Figure: Difference between optimizing for Mean Cost vs. CVaR.

Sauté RL: Methodology

Almost Sure Safety [Sootla et al., 2022]

State Augmentation

The most robust way to ensure safety is to let the policy "know" how much safety budget it has left.

Mechanism

1. Augment State: $\tilde{s}_t = [s_t, z_t]$.
2. z_t represents the "remaining safety budget."
3. Dynamics: $z_{t+1} = (z_t - c_t)/\gamma$.

Objective Reshaping

$$\tilde{r}(\tilde{s}_t, a_t) = \begin{cases} r(s_t, a_t) & \text{if } z_t > 0 \\ -\infty & \text{if } z_t \leq 0 \end{cases}$$

Sauté RL: Intuition & Results

Why "Sauté"? Safety AUgmenTEd MDP.

Resulting Behavior

The constraint is baked into the value function. If the agent sees z_t getting low, the value function drops, and it naturally avoids risky actions to preserve the budget.

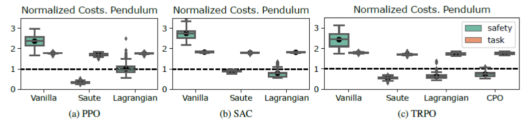

Figure 2. Sauté RL as a plug-n-play approach. Box plots for normalized safety(on the left) and task (on the right) costs for SAC, PPO and TRPO-type algorithms on pendulum swing-up environment with the safety budget 30 after 300 epochs of training. In all figures the task cost are divided by -100 and the safety costs are divided by 30, the dashed lined indicate the safety threshold. In all cases, “sautéed” algorithms deliver safe policies with probability 1 (outliers whiskers do not cross the dashed line), while Lagrangian methods and CPO have trajectories violating the safety constraints. For task costs the higher values are better.

Figure: Sauté RL effectively eliminates constraint violations (Zero violations shown in box plots).

Synthesis: The Evolution of Safe RL

1. **Unconstrained (TRPO):** Stable updates, but no safety.
2. **Primal-Dual (RCPO):** Adds constraints via penalty. Good asymptotically, unstable during training.
3. **Primal (IPO/CRPO):** Fixes instability. Direct switching or barriers are easier to tune than Lagrange multipliers.
4. **Structural (Sauté/WCSAC):** Redefines the problem.
    - Expectations are insufficient.
    - We need **Tail Risk** (WCSAC) or **Hard Budgets** (Sauté) to be truly safe.

Class Discussion

Topic 1: Reward Shaping vs. Hard Constraints

Sauté RL modifies the reward to $-\infty$ upon violation. In standard RL, huge negative rewards can destroy exploration. Why does Sauté RL work where simple reward shaping usually fails?

Topic 2: Tuning Difficulty

CRPO switches gradients. IPO uses a log barrier. RCPO uses a lagrange multiplier. Which hyperparameter do you think is intuitively harder to tune for a new environment, and why?

Topic 3: Defining Safety

Imagine you are deploying a robot in a hospital. Would you use an Expected Cost constraint (CPO/RCPO) or a Tail Risk constraint (WCSAC)? Is "Expected Safety" ever acceptable in the real world?