Trust Region Policy Optimization

By: John Schulman et. al.

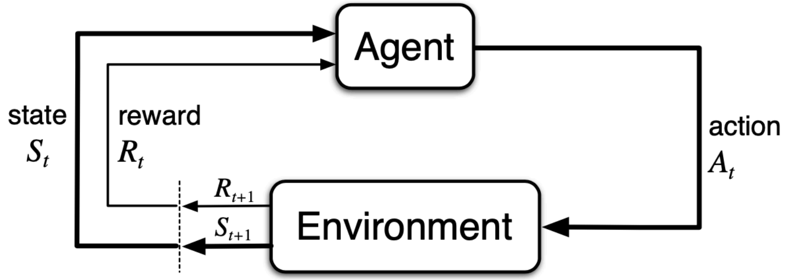
Presented by: Jonathan Salfity
September 26, 2023

Outline

- Reinforcement Learning and Policy Optimization
- Policy Optimzation: Refresher
- Trust Region *applied* to Policy Optimization
- Trust Region Policy Optimization: Theoretical and Implementation
- Experimental Results
- Critique / Limitations
- Extended Readings
- Summary

Context: Reinforcement Learning

We cast an agent's decision-making framework in a Reinforcement Learning setting:



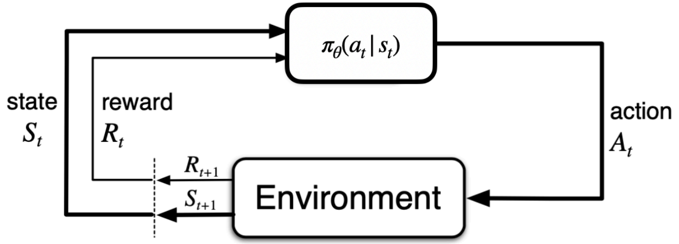
The goal of the agent is to take actions that maximizes the expected sum of future rewards:

$$\text{goal(agent)} = \max \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right]$$

http://incompleteideas.net/book/RLbook2020.pdf

Motivation: Policy Optimization within RL

In **policy optimization**, an agent will be represented by a policy, $\pi : S \mapsto A$, parameterized by θ.



In **policy optimization**, an agent wants to directly learn parameters, θ, such that the choice in parameters result in collecting high rewards:

$$\text{goal(agent)} = \max_\theta \eta(\pi(a_t | s_t, \theta)) = \max_\theta \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t)\Big]$$

An example of Policy Optimization

https://guardianbikes.com/blogs/around-the-block/how-to-teach-a-kid-to-ride-a-bike

When a child learns to ride a bicycle, they don't care about the underlying dynamics of the bicyle…

…they only cares about controlling their hands and feet to to keep balance and not fall.

Policy Optimization: Notation and Useful Expressions

Functions to help better understand policy optimization by assessing the **quality of actions** w.r.t their underlying parameters, θ, and the objective function, $\eta()$.

$$s_{t+1} \sim P(s_{t+1} \mid s_t, a_t)$$

Probabilistic transition dynamics

$$a_t \sim \pi(a_t \mid s_t; \theta)$$

Actions conditioned on the state and parameterized by θ

$$\eta(\pi_\theta) = \mathbb{E}_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right]$$

The discounted sum of future rewards

$$V_\pi(s_t) = \mathbb{E}_{a_t, s_{t+1}, \ldots}\left[\sum_{l=0}^{\infty} \gamma^t r(s_{l+1})\right]$$

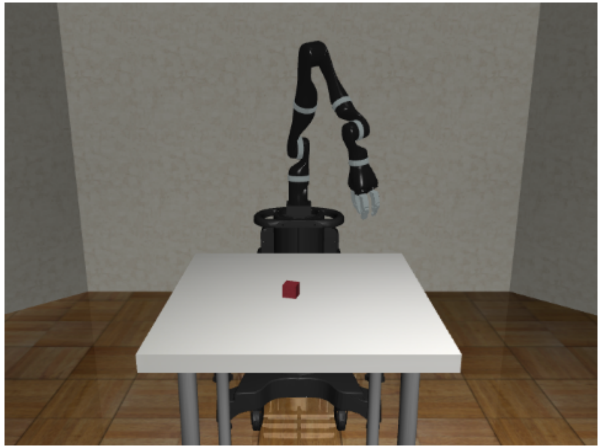
The value of a state

$$Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots}\left[\sum_{l=0}^{\infty} \gamma^t r(s_{l+1})\right]$$

The value of a state-action pair

$$\boxed{A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)}$$

The advantage function quantifies the marginal improvement of taking a particular action against any action.



Snapshot of a manipulator during an episode.

Goal: *Move towards the object*

Reward: *End-effector distance to the object*

States: *Joint angles*

Actions: *Commanded joint torques*

Policy Optimization: Refresher

Let's focus on this optimization problem:

$$\max_{\theta} \eta(\pi(a_t | s_t, \theta)) = \max_{\theta} \eta(\pi_\theta)$$

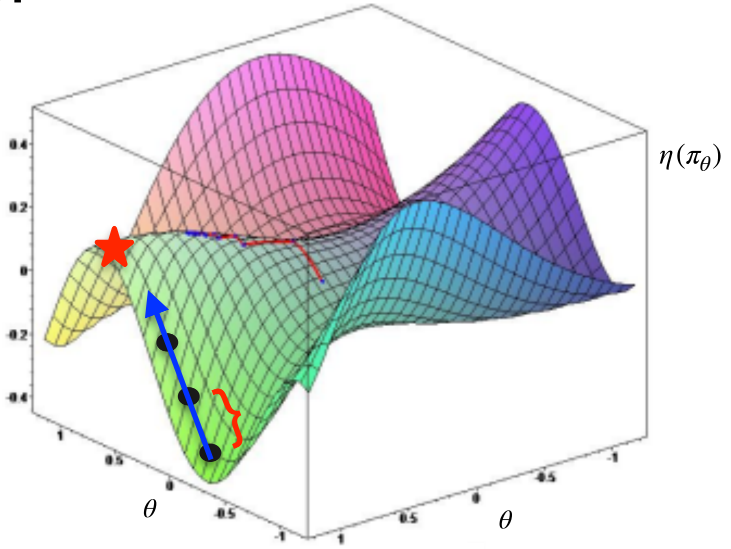
What is the **maximum** of the function?

$$\eta(\pi^*_\theta) \geq \eta(\pi_\theta)$$

How do we **find** the maximum of a function?

Iterate	Parameter	Function Value
k	θ_k	$\eta(\pi_{\theta_k})$
$k+1$	θ_{k+1}	$\eta(\pi_{\theta_{k+1}})$
$k+2$	θ_{k+2}	$\eta(\pi_{\theta_{k+2}})$
$\dots$	$\dots$	$\dots$
$??$	θ^*	$\eta(\pi^*_\theta)$

$\eta(\pi_{\theta_{k+1}}) \geq \eta(\pi_{\theta_k})$



Parameter Update Equation:

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta \eta(\pi_{\theta_k})$$

*The **step-size**, α, determines how far to move from the last iterate.*

*The **gradient** gives us directional information.*

Policy Optimization: Issues with Gradients, Step-Size

Optimization problem:

$$\max_{\theta} \eta(\pi_\theta)$$

(Theoretical) Parameter Update equation:

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta \eta(\pi_{\theta_k})$$

How do we **numerically compute** the gradient?

$$\widehat{\nabla_\theta \eta(\pi_\theta)} = \mathbb{E}_{\tau \sim \pi_\theta}\Big[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \text{log}\pi_\theta(a_t \mid s_t) A_\pi(s_t, a_t)\Big]$$

***Policy Gradient Theorem:** We can use data generated by π_θ to form an estimate of the gradient.*

(Numerical) Parameter Update equation:

$$\theta_{k+1} \leftarrow \theta_k + \alpha \widehat{\nabla_\theta \eta(\pi_\theta)}$$

Issues:

*The **gradient estimate,** $\widehat{\nabla_\theta \eta(\pi_\theta)}$, is only a first order approximation, i.e. the function is assumed to be linear, curvature information is not captured, etc.*

Globally estimating the gradient can have high variance with large state and action spaces.

*The **step-size**, α, is relative to the parameters, θ, and not the objective function, $\eta(\pi_\theta)$, nor the policy π_θ.*

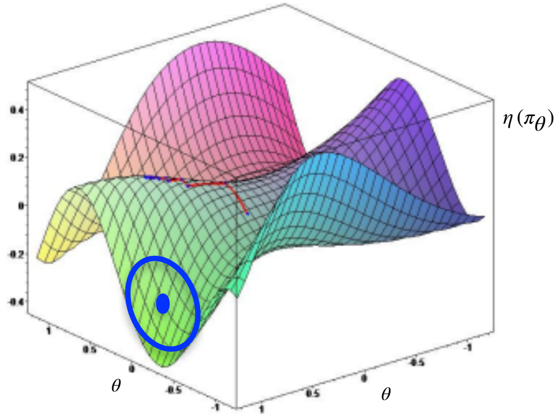
Large step-sizes result in overshooting. Small step-size result in slow performance.

Is there a better method for the update equation?

Trust Region *applied* to Policy Optimization

Define a local, **trust region**, to be a region we think we understand better.
Find the optimal point within this region.

$$\theta_{k+1} = \arg\max_{\theta} \mathscr{L}(\theta_k, \theta)$$
$$\text{s.t. } ||\alpha|| \leq \delta$$



Trust Region Policy Optimization (TRPO)

TRPO introduces an iterative procedure for optimizing policies with guaranteed monotonic improvements.

$$\theta_{k+1} \leftarrow \theta_k + \text{update}$$

$$\implies \eta(\pi_{\theta_{k+1}}) \geq \eta(\pi_{\theta_k})$$

Parameter update rule guarantees the next policy iterate is better than the last.

First, the authors prove that minimizing a **surrogate objective function** guarantees policy improvement with non-trivial step size.

$$\theta_{k+1} = \arg\max_{\theta} \mathscr{L}(\theta_k, \theta)$$

$$D_{KL}(\pi_\theta || \pi_{\theta_k}) \leq \delta$$

Surrogate loss, $\mathscr{L}(\theta_k, \theta)$, represents a local region we understand better and is simpler to optimize.

Average KL-divergence between policies across states visited by the old policy.

Next, the authors use **numerical approximations** to yield a practical algorithm, namely TRPO.

$$\mathscr{L}(\theta_k, \theta) = \mathbb{E}_{s,a\sim\pi_{\theta_k}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A_{\pi_{\theta_k}}(s,a)\right]$$

$$\bar{D}_{KL}(\theta || \theta_k) := D_{KL}(\pi_\theta || \pi_{\theta_k})$$

The surrogate loss is computed by comparing the marginal benefit of a new policy w.r.t to an old policy.

The distance between parameters is essentially the same as the distance between policies.

Trust Region Policy Optimization (Theoretical)

TRPO introduces an iterative procedure for optimizing policies with guaranteed monotonic improvements.

$$\theta_{k+1} \leftarrow \theta_k + \text{update}$$

$$\implies \eta(\pi_{\theta_{k+1}}) \geq \eta(\pi_{\theta_k})$$

Parameter update rule guarantees the next policy iterate is better than the last.

First, the authors prove that minimizing a **surrogate objective function** guarantees policy improvement with non-trivial step size.

$$\theta_{k+1} = \arg\max_{\theta} \mathscr{L}(\theta_k, \theta)$$

$$D_{KL}(\pi_\theta || \pi_{\theta_k}) \leq \delta$$

Surrogate loss, $\mathscr{L}(\theta_k, \theta)$, that is simpler to optimize.

Average KL-divergence between policies across states visited by the old policy

Next, the authors use **numerical approximations** to yield a practical algorithm, namely TRPO.

$$\mathscr{L}(\theta_k, \theta) = \mathbb{E}_{s,a\sim\pi_{\theta_k}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A_{\pi_{\theta_k}}(s,a)\right]$$

$$\bar{D}_{KL}(\theta || \theta_k) = D_{KL}(\pi_\theta || \pi_{\theta_k})$$

The surrogate loss is computed by comparing the marginal benefit of a new policy w.r.t to an old policy

The distance between parameters is equivalent to the distance between policies.

Trust Region Policy Optimization (Theory)

We need a notion of **improvement** between policies:

$$\tilde{\pi}, \quad \eta(\tilde{\pi})$$
$$\pi, \quad \eta(\pi)$$

Suppose we have two policies. π and $\tilde{\pi}$ and two corresponding reward function values.

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

*We can compare the difference between two policies associated reward function values using the **advantage function.***

$$\sum_a \tilde{\pi}(a \mid s) A_\pi(s,a) \geq 0$$
$$\implies \eta(\tilde{\pi}) \geq \eta(\pi)$$

We know there will be improvement if this holds.

$$\mathscr{L}_\pi(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a \mid s) A_\pi(s,a)$$

*However, due to the large state and action spaces, we find a **local approximation of the loss function w.r.t. the two policies.***

Which represents our "surrogate loss".

Measures the relative difference between rewards gathered with $\tilde{\pi}$ in comparison to rewards gathered with π.

But there lies a problem because this value is not applicable to general stochastic policy classes

Trust Region Policy Optimization (Theory)

Now with a stochastic, parameterized policies, we need a notion for the difference between policies, $\pi_\theta, \pi_{\tilde{\theta}}$, w.r.t the parameterization, $\theta, \tilde{\theta}$:

$$D_{KL}(\theta || \tilde{\theta}) := D_{KL}(\pi_\theta || \pi_{\tilde{\theta}})$$

KL Divergence between parameters, θ, is the same as distance between policy, π_θ

$$D_{KL}(\pi_\theta || \pi_{\tilde{\theta}}) \leq \delta$$

Define some maximum distance between the two policies

$$\boxed{\begin{gathered}\theta_{k+1} = \arg\max_\theta \mathscr{L}(\theta_k, \theta) \\ \text{s.t. } D_{KL}(\pi_\theta || \pi_{\theta_k}) \leq \delta\end{gathered}}$$

Use this distance as a constraint in our optimization problem.

TRPO Theoretical update.

Trust Region Policy Optimization (TRPO)

TRPO introduces an iterative procedure for optimizing policies with guaranteed monotonic improvements.

$$\theta_{k+1} \leftarrow \theta_k + \text{some improvement}$$

$$\implies \eta(\pi_{\theta_{k+1}}) \geq \eta(\pi_{\theta_k})$$

Parameter update rule guarantees the next policy iterate is better than the last.

First, the authors prove that minimizing a **surrogate objective function** guarantees policy improvement with non-trivial step size.

$$\theta_{k+1} = \arg\max_{\theta} \mathscr{L}(\theta_k, \theta)$$

$$D_{KL}(\pi_\theta || \pi_{\theta_k}) \leq \delta$$

Surrogate loss, $\mathscr{L}(\theta_k, \theta)$, that is simpler to optimize.

Average KL-divergence between policies across states visited by the old policy

Next, the authors use **numerical approximations** to yield a practical algorithm, namely TRPO.

$$\mathscr{L}(\theta_k, \theta) = \mathbb{E}_{s,a\sim\pi_{\theta_k}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A_{\pi_{\theta_k}}(s,a)\right]$$

$$\bar{D}_{KL}(\theta || \theta_k) = D_{KL}(\pi_\theta || \pi_{\theta_k})$$

The surrogate loss is computed by comparing the marginal benefit of a new policy w.r.t to an old policy

The distance between parameters is equivalent to the distance between policies.

Trust Region Policy Optimization (Implementation)

We need a ***practical*** method to relate the ***relative goodness*** of two policies.

Importance Sampling computes the marginal benefit of a new policy with respect to an old policy.

$$\mathscr{L}(\theta_k, \theta) = \mathbb{E}_{s,a \sim \pi_{\theta_k}} \Big[\frac{\boxed{\pi_\theta(a \mid s)}}{\boxed{\pi_{\theta_k}(a \mid s)}} A_{\pi_{\theta_k}}(s, a) \Big]$$

Directly comparing a new policy with data from the old policy.

Key Insight: *The objective function is rewritten using samples from an old policy.*

Now, for each iterate, we can compute optimal next parameter by solving:

$$\boxed{\begin{aligned} \theta_{k+1} &= \arg\max_\theta \mathbb{E}_{s,a \sim \pi_{\theta_k}} \Big[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)} A_{\pi_{\theta_k}}(s, a) \Big] \\ \text{s.t. } & D_{KL}(\pi_\theta \,||\, \pi_{\theta_k}) \leq \delta \end{aligned}}$$

The surrogate objective represents the marginal improvement by a policy w.r.t an old policy.

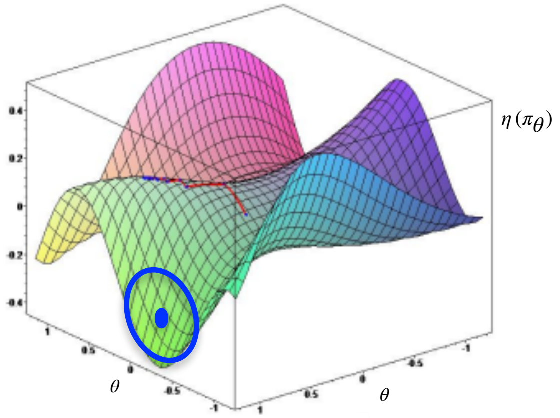
The constraint measures the distance between two policies.

Trust Region Policy Optimization (Implementation)

Main Idea:

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}(\theta_k, \theta)$$

$$\text{s.t. } D_{KL}(\pi_\theta || \pi_{\theta_k}) \leq \delta$$



Algorithm Implementation:

Actually, the author's work around the work around…

Appendix C: Efficiently Solving the Trust-Region Constrained Optimization Problem

Taylor Expansion of the objective and the constraint around θ_k

$$\mathcal{L}(\theta_k, \theta) \approx \widehat{\nabla_\theta \eta(\pi_\theta)}^\top (\theta - \theta_k)$$ *Linear Approximation*

$$D_{KL}(\pi_\theta || \pi_{\theta_k}) \approx \frac{1}{2}(\theta - \theta_k)^\top H(\theta - \theta_k) \leq \delta$$ *Quadratic Approximation*

TRPO Algorithm Sketch

```
for k = 0, 1, 2, do:
1.  Collect trajectories by running policy $\pi_{\theta_k}$
2.  Compute rewards-to-go and advantage estimate
3.  Estimate the policy gradient
4.  Use Conjugate Gradient method to compute search direction
5.  Update the policy with backtracking line search
6.  Fit the value function by regression on MSE
```

The constraint is solved through a Quadratic Program, which means we have an exact solution at the cost of inverting a Hessian matrix, H.

https://spinningup.openai.com/en/latest/algorithms/trpo.html

Proximal Policy Optimization (PPO)

Motivation: Take the biggest possible improvement step on a policy within a local, “trusted” region.

TRPO:

Theoretical update equation is optimizing in a local region:

$$\theta_{k+1} = \arg\max_{\theta} \mathscr{L}(\theta_k, \theta)$$

$$\text{s.t. } D_{KL}(\pi_\theta || \pi_{\theta_k}) \leq \delta$$

Practical Implementation:

$$\mathscr{L}(\theta_k, \theta) \approx \widehat{\nabla_\theta \eta(\pi_\theta)}^\top (\theta - \theta_k)$$

$$D_{KL}(\pi_\theta || \pi_{\theta_k}) \approx \frac{1}{2}(\theta - \theta_k)^\top H(\theta - \theta_k) \leq \delta$$

***The main problem** lies in numerically computing the Quadratic Program, i.e. the second order constraint.*

***The solution** is to use a clipping function to constrain the distance beween the two policies.*

PPO:

Theoretical update equation is optimizing in a local region:

$$\theta_{k+1} = \arg\max_{\theta} \mathscr{L}^{CLIP}(\theta_k, \theta)$$

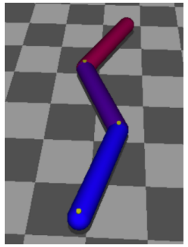
$$\mathscr{L}^{CLIP}(\theta_k, \theta) = \min\left(\frac{\pi_\theta}{\pi_{\theta_k}} A_{\pi_{\theta_k}}(s,a), \text{clip}(1-\epsilon, 1+\epsilon) A_{\pi_{\theta_k}}(s,a)\right)$$

In plain English, PPO uses no formal constraints and instead clips the distance between policies in the loss function.

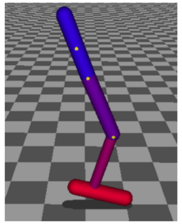
… i.e. PPO doesn’t have to solve a Quadratic Program for the constraint.

Experimental Setup: Simulated Robotic Locomotion

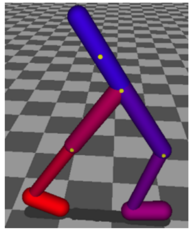
Swimmer
- **Observation:** 10-d state space
- **Reward:** linear reward for forward progress, quadratic penalty on joint effort
- **Action:** Torque applied to rotors



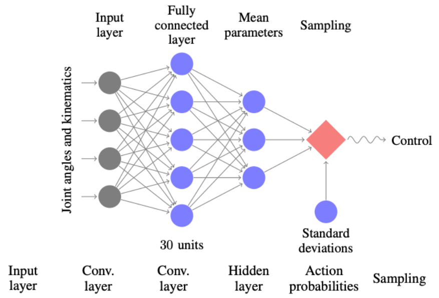
Hopper
- **Observation:** 12-d state space
- **Reward:** linear reward for forward progress, quadratic penalty on joint effort
- **Action:** Torque applied to rotors



Walker
- **Observation:** 18-d state space
- **Reward:** linear reward for forward progress, quadratic penalty on joint effort. Penalty for strong impacts.
  Action: Torque applied to rotors



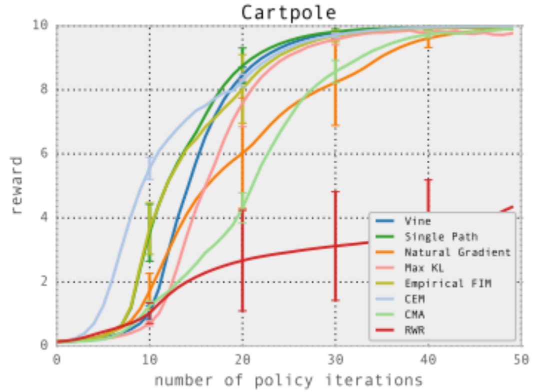
Neural Network Architecture:

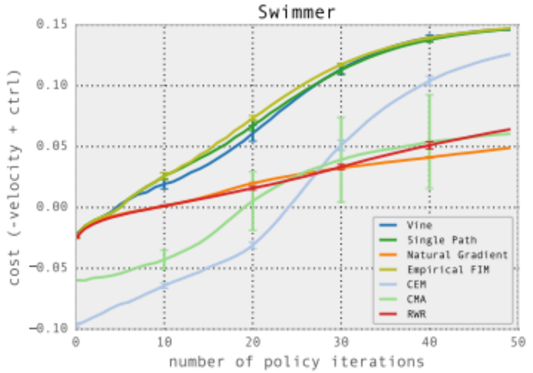


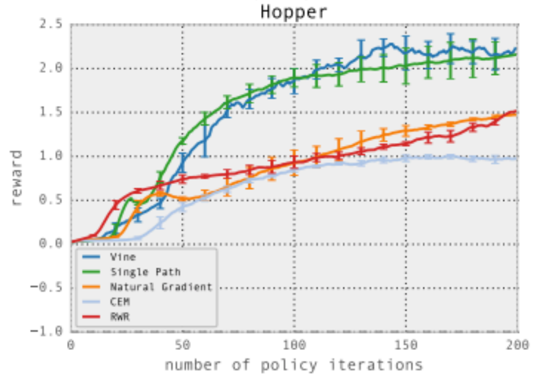
KL Divergence constraint hardcoded:

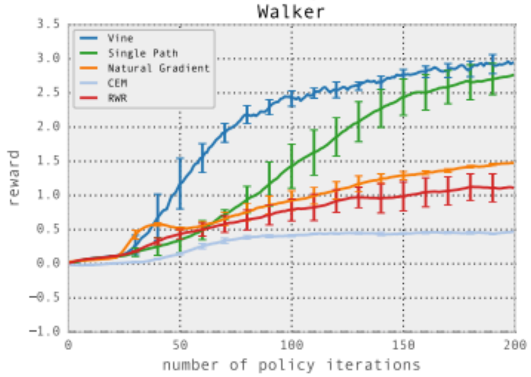
$$D_{KL}^{\max}(\theta_{\text{old}}, \theta) \leq 0.2$$

Experimental Results: Simulated Robotic Locomotion









Learning curves contrast the two TRPO implementations against previous methods.

Key Insights:

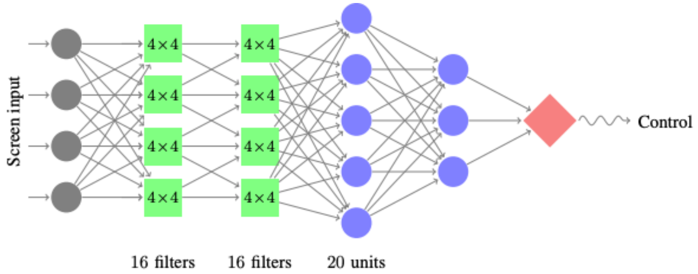
- TRPO achieves the largest reward with a steady increase.
- TRPO Vine method shows low variance throughout policy iterations.

Questions:

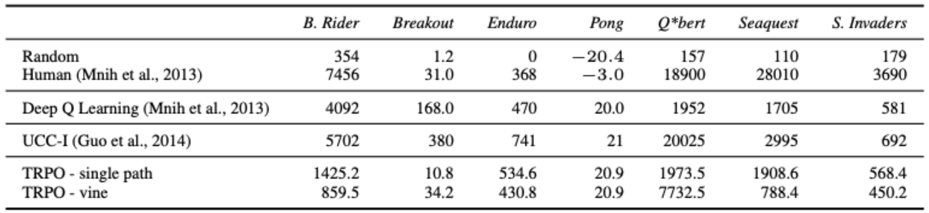
- Why did they choose this NN architecture?
- Why did 0.2 for the KL Divergence constraint?
- How much compute time and resources is the local optimization consuming?

Experimental Setup: Playing Games from Images

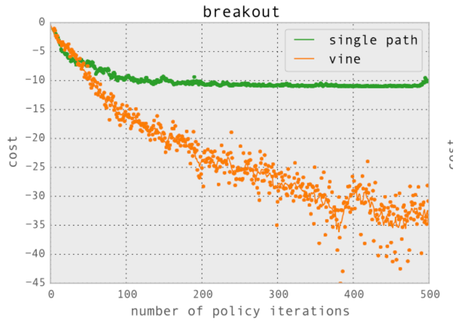
Atari games use non-stationary, raw images as input. Policies must learn complex sequences of behaviors and manage delayed rewards.



Experimental Results: Playing Games from Images



	B. Rider	*Breakout*	*Enduro*	*Pong*	*Q*bert*	*Seaquest*	*S. Invaders*
Random	354	1.2	0	-20.4	157	110	179
Human (Mnih et al., 2013)	7456	31.0	368	-3.0	18900	28010	3690
Deep Q Learning (Mnih et al., 2013)	4092	168.0	470	20.0	1952	1705	581
UCC-I (Guo et al., 2014)	5702	380	741	21	20025	2995	692
TRPO - single path	1425.2	10.8	534.6	20.9	1973.5	1908.6	568.4
TRPO - vine	859.5	34.2	430.8	20.9	7732.5	788.4	450.2



Key Insights:

- TRPO is not always better in this case, which may imply TRPO is better for continuous control examples.

What's missing:

- Statistics explaining the reward variance over games and other error statistics.
- An explanation why the vine method never improves.

Critique / Limitations / Open Issues

- How computationally expensive is the Quadratic Program necessary to solve the constraint?
- How much data is required to form function estimates?
- How do we form intution on policy distance constraint? Can we alternatively use an adaptive distance?
- How does the trust-region constraint impact the trade-off between exploration and exploitation?
- From the results, TRPO seems to only work on continuous control tasks, why?

Extended Readings

- TRPO is a very established and well-trusted algorithm. There exists many sources to explain the theory and implementation: Pieter Abeel Deep RL video, Spinning Up from OpenAI, Medium Article.

- Trust Regions are not invented by TRPO! They exist in nonlinear optimization literature.

- And of course, PPO is the follow-up paper to TRPO.

Summary

- TRPO proposes a trust region method for optimizing stochastic control policies
- The theoretical contributions include proving monotonic increase in policy optimization through a trust region method.
- The practical contributions show how to approximate the necessary functions with collected data.
- Experimentation show better results for continuous control tasks, where inputs are low-dimensional states rather than pixels.