

Reliable RL: An Algorithmic Perspective

Logan Engstrom

(with A. Ilyas*, S. Santurkar, D. Tsipras, F. Janoos, L.
Rudolph, and A. Madry)



gradsci.org



Reinforcement Learning (RL)



Reinforcement Learning (RL)



Reinforcement Learning (RL)

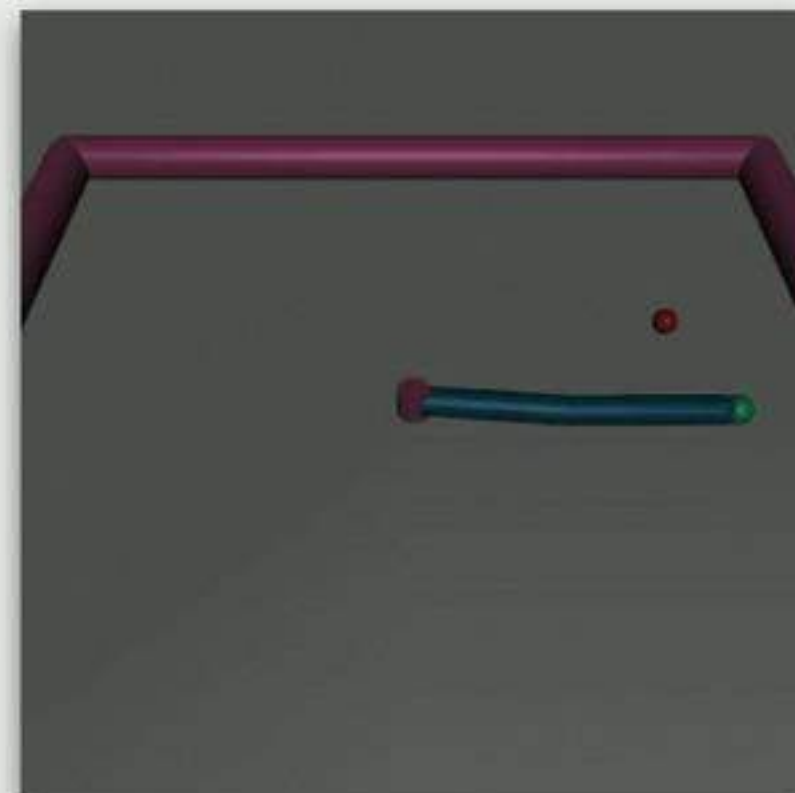
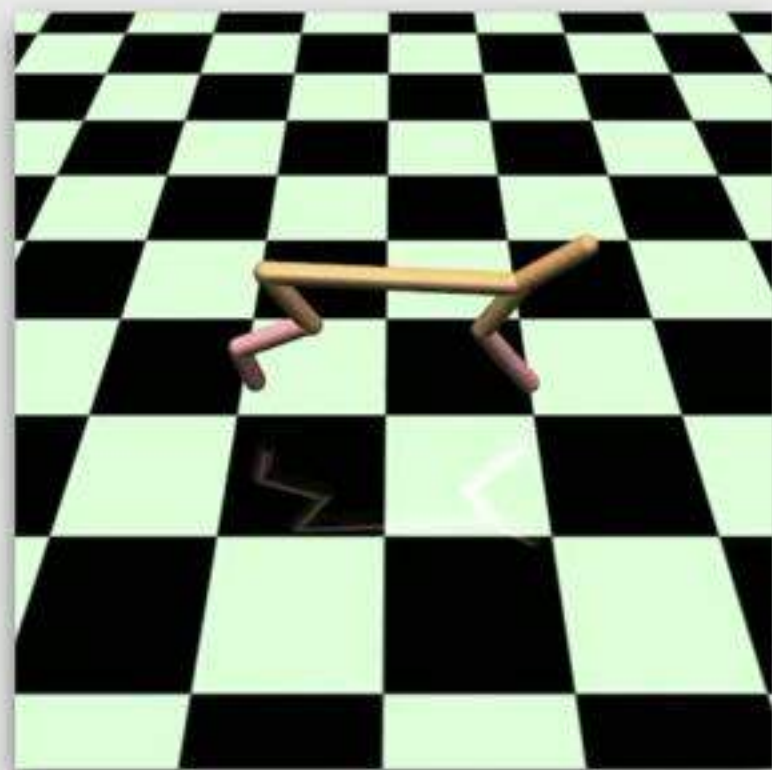


Is RL “real-world-ready”?

Is RL “real-world-ready”?

(Spoiler: No)

Deep RL is unreliable even in simple settings...



How do we get reliable RL?

An **algorithmic** understanding of
modern RL methods

The RL Setup

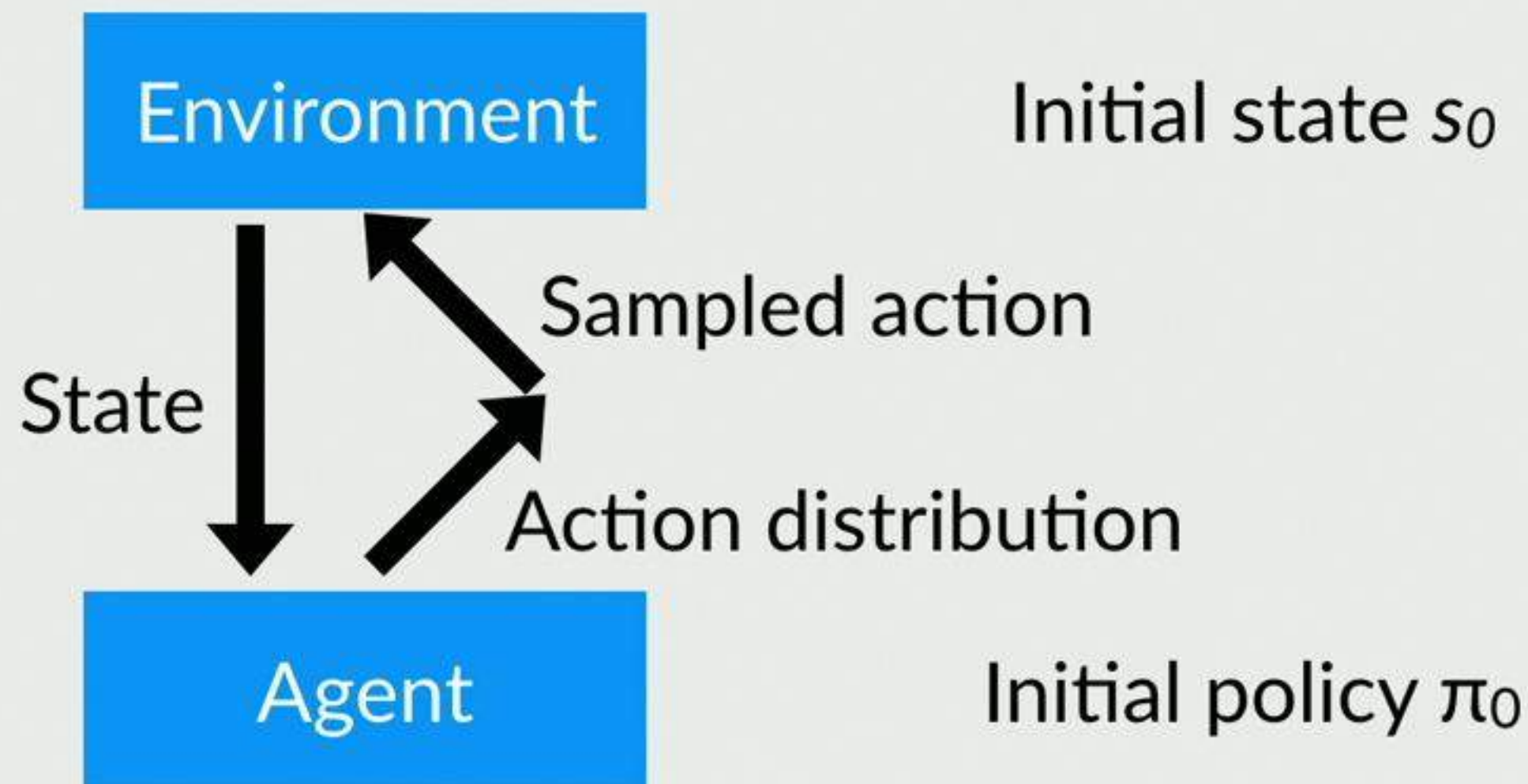
Environment

Initial state s_0

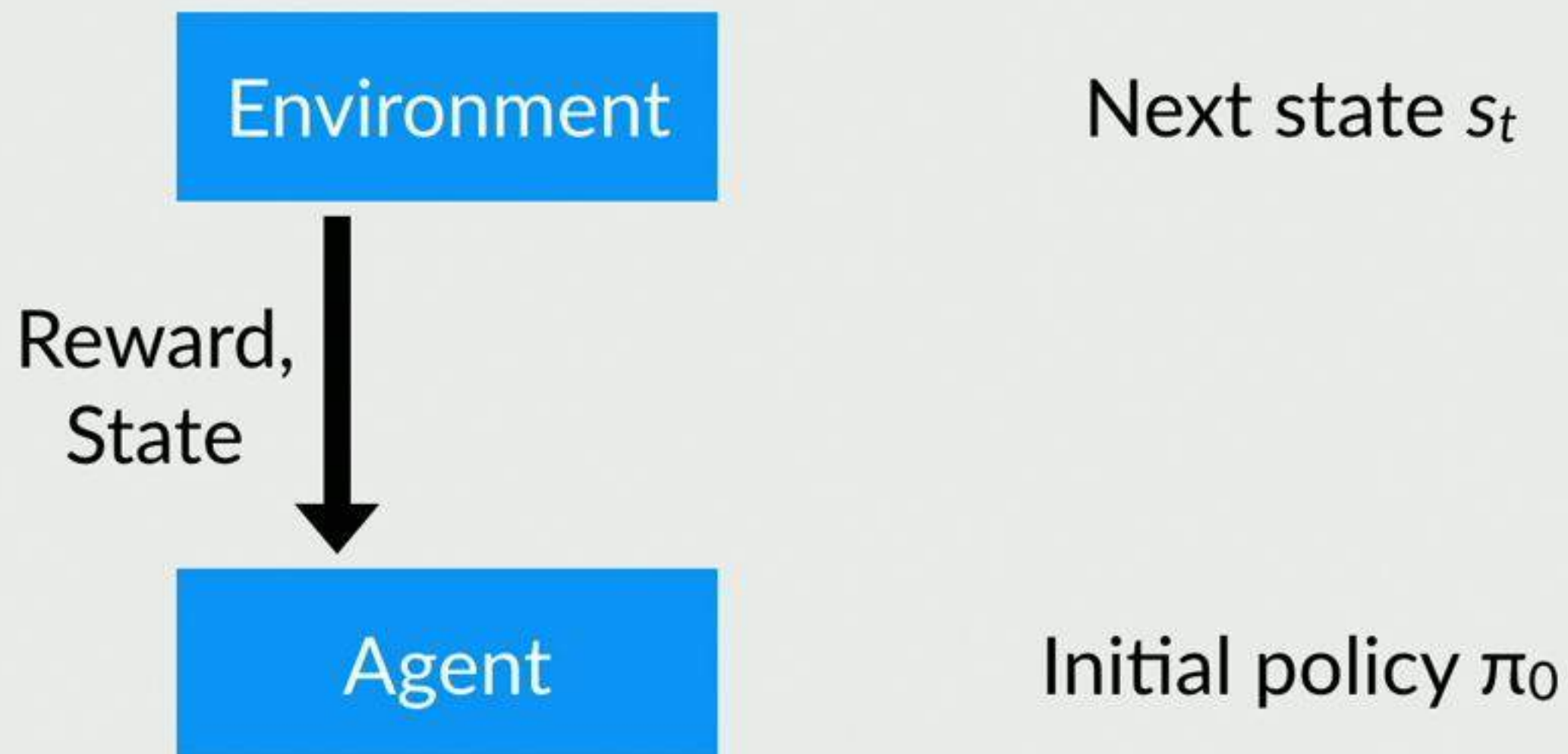
Agent

Initial policy π_0

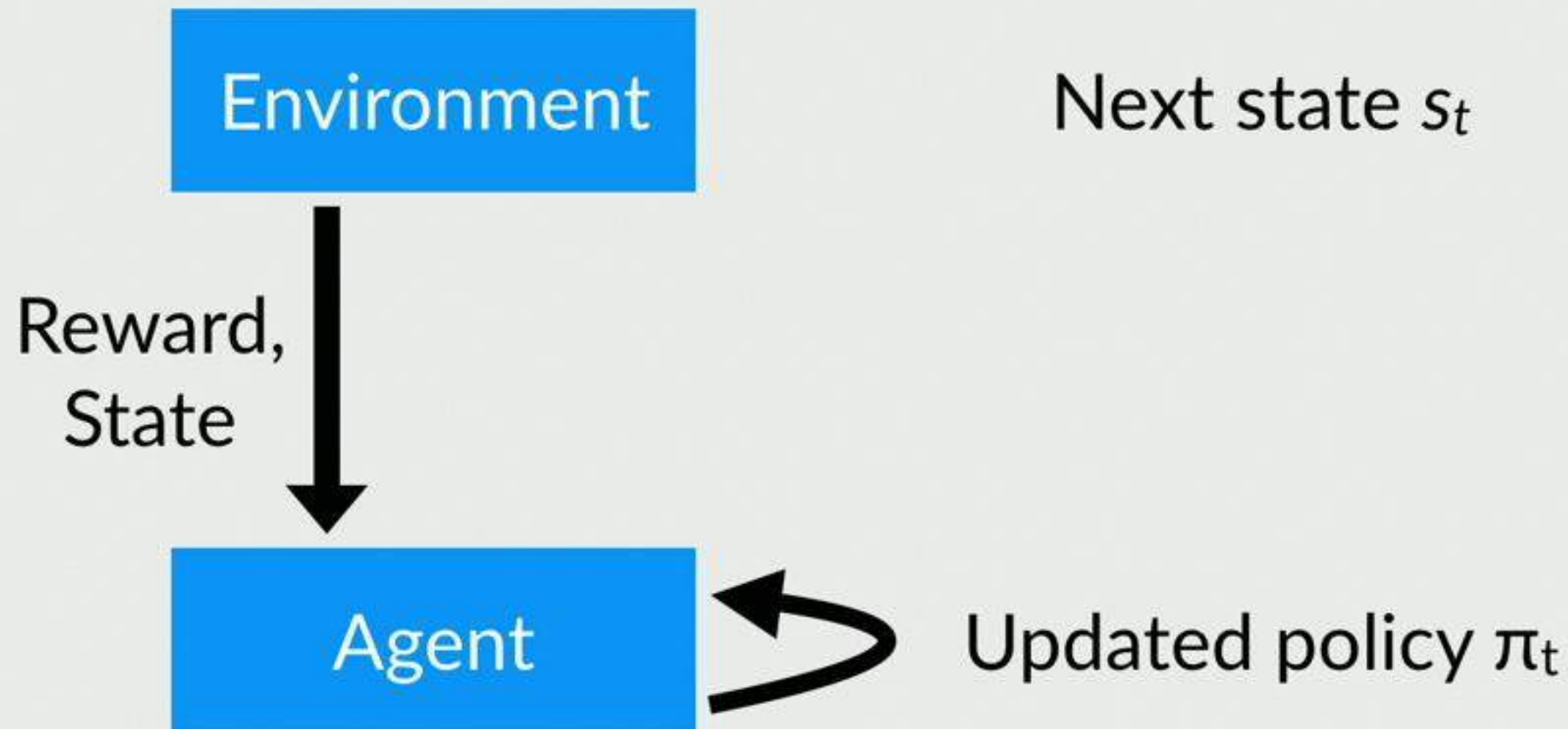
The RL Setup



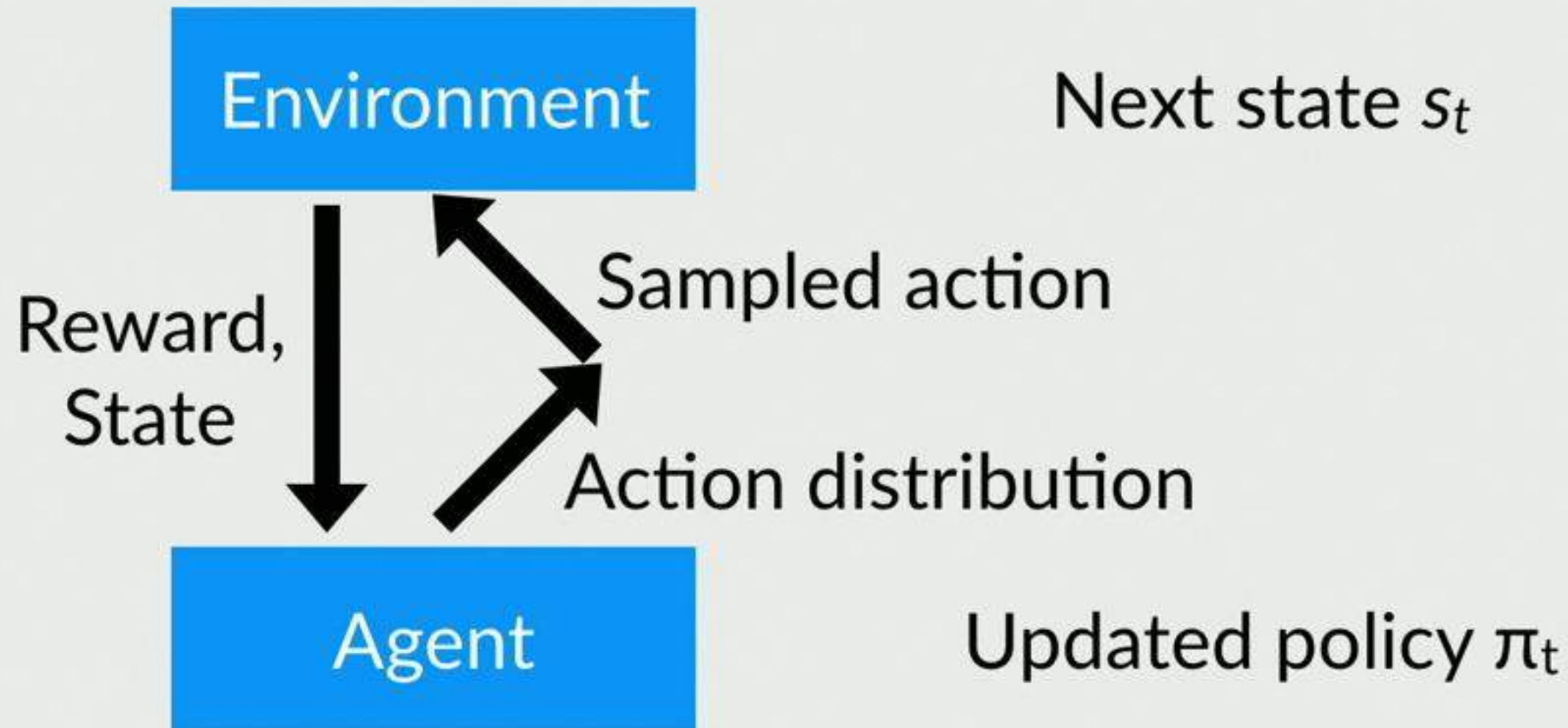
The RL Setup



The RL Setup



The RL Setup



Goal: Maximize expected total reward

(over trajectories)

Policy Gradient Algorithms

Policy Gradients

Key Principle: View our goal as an optimization problem

$$\theta^* = \arg \max_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{(s,a) \in \tau} r(s,a) \right]$$

Policy Gradients

Key Principle: View our goal as an optimization problem

$$\theta^* = \arg \max_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{(s,a) \in \tau} r(s,a) \right]$$

The diagram illustrates the equation for finding the optimal policy parameters θ^* . It features two blue circles and two arrows. The first blue circle highlights the expectation operator $\mathbb{E}_{\tau \sim \pi_{\theta}}$, with an arrow pointing to the text "Expected value (over sampled trajectories) under current policy". The second blue circle highlights the summation term $\sum_{(s,a) \in \tau} r(s,a)$, with an arrow pointing to the text "Total reward".

Expected value (over sampled trajectories) under current policy

Total reward

Policy Gradients

Key Principle: View our goal as an optimization problem

$$\theta^* = \arg \max_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{(s,a) \in \tau} r(s,a) \right]$$

Method of choice: gradient descent

Policy Gradients

Key Principle: View our goal as an optimization problem

$$\theta^* = \arg \max_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{(s,a) \in \tau} r(s,a) \right]$$

No gradient access

Method of choice: gradient descent

Policy Gradients

Can we instead get a good *estimate* of the gradient?

$$\nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{(s,a) \in \tau} r(s,a) \right] = ???$$

Policy Gradients

Can we instead get a good *estimate* of the gradient?

$$\nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{(s,a) \in \tau} r(s,a) \right] = \mathbb{E}_{\tau \sim \pi_{\theta}} [g(\tau)]$$

The Policy Gradient

Policy Gradients

Can we instead get a good *estimate* of the gradient?

$$\nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{(s,a) \in \tau} r(s,a) \right] = \mathbb{E}_{\tau \sim \pi_{\theta}} [g(\tau)]$$

$$\approx \frac{1}{N} \sum_{\tau \sim \pi_{\theta}} [g(\tau)]$$

Policy Gradients

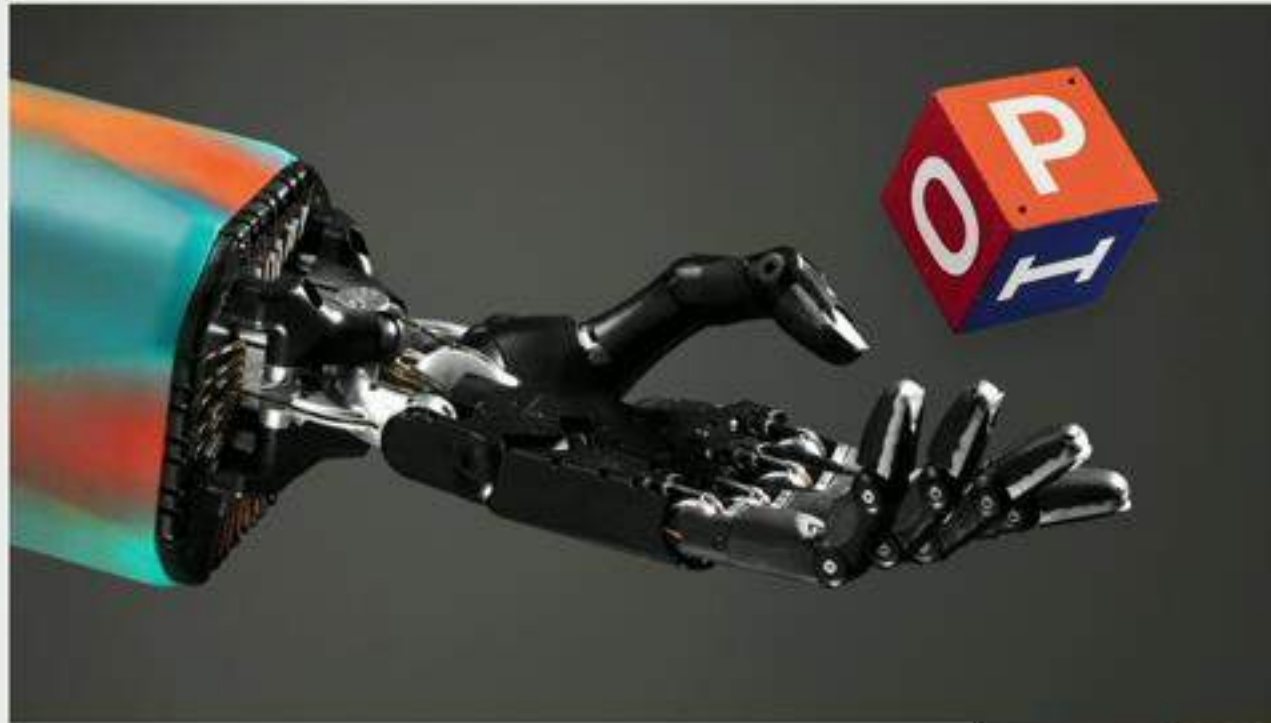
Can we instead get a good *estimate* of the gradient?

$$\nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{(s,a) \in \tau} r(s,a) \right] = \mathbb{E}_{\tau \sim \pi_{\theta}} [g(\tau)]$$

$$\approx \frac{1}{N} \sum_{\tau \sim \pi_{\theta}} [g(\tau)]$$

Then: use estimate in gradient descent!

Policy Gradient Successes



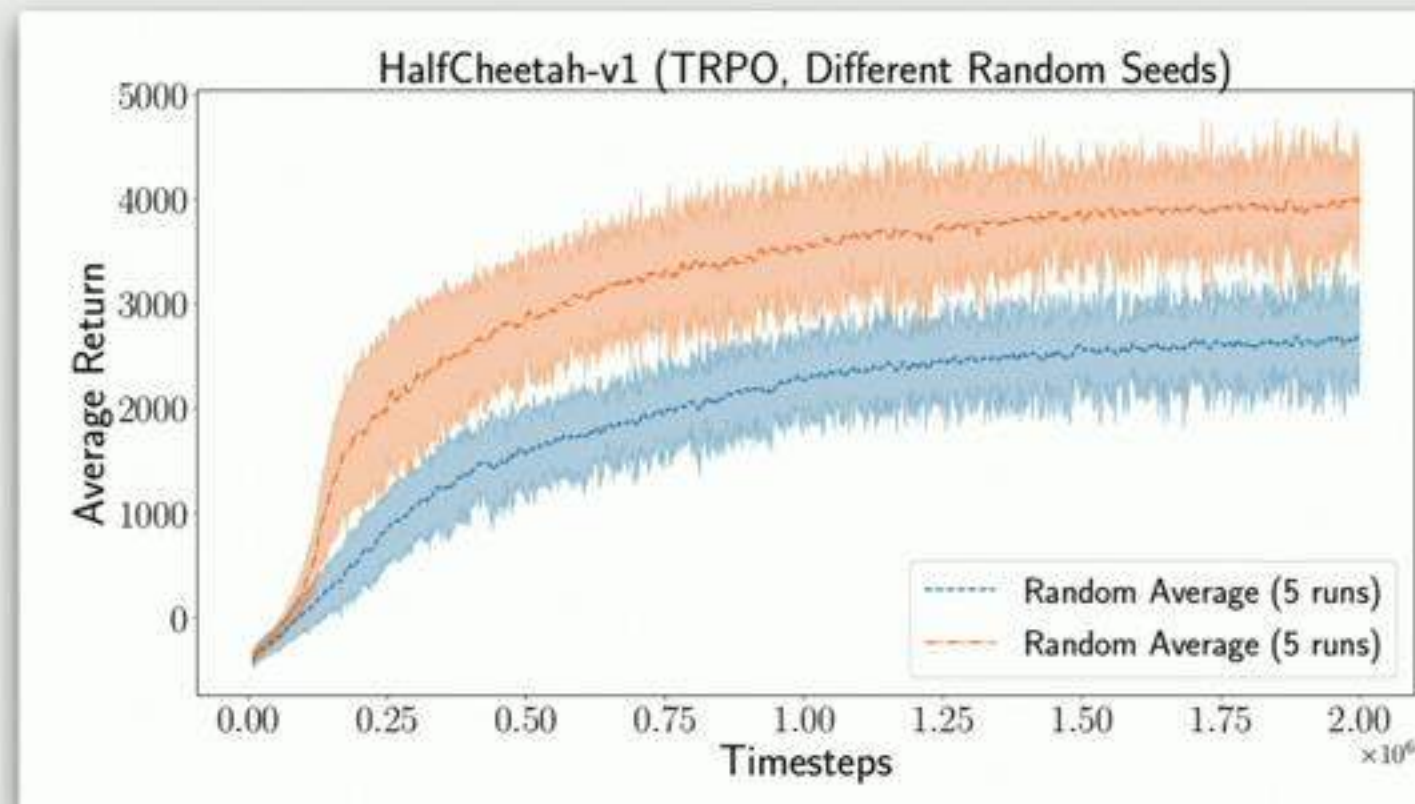
The Rotten Truth of Deep RL



The Rotten Truth of Deep RL

Deep RL can successfully solve tasks, but has...

- Poor reliability over repeated runs

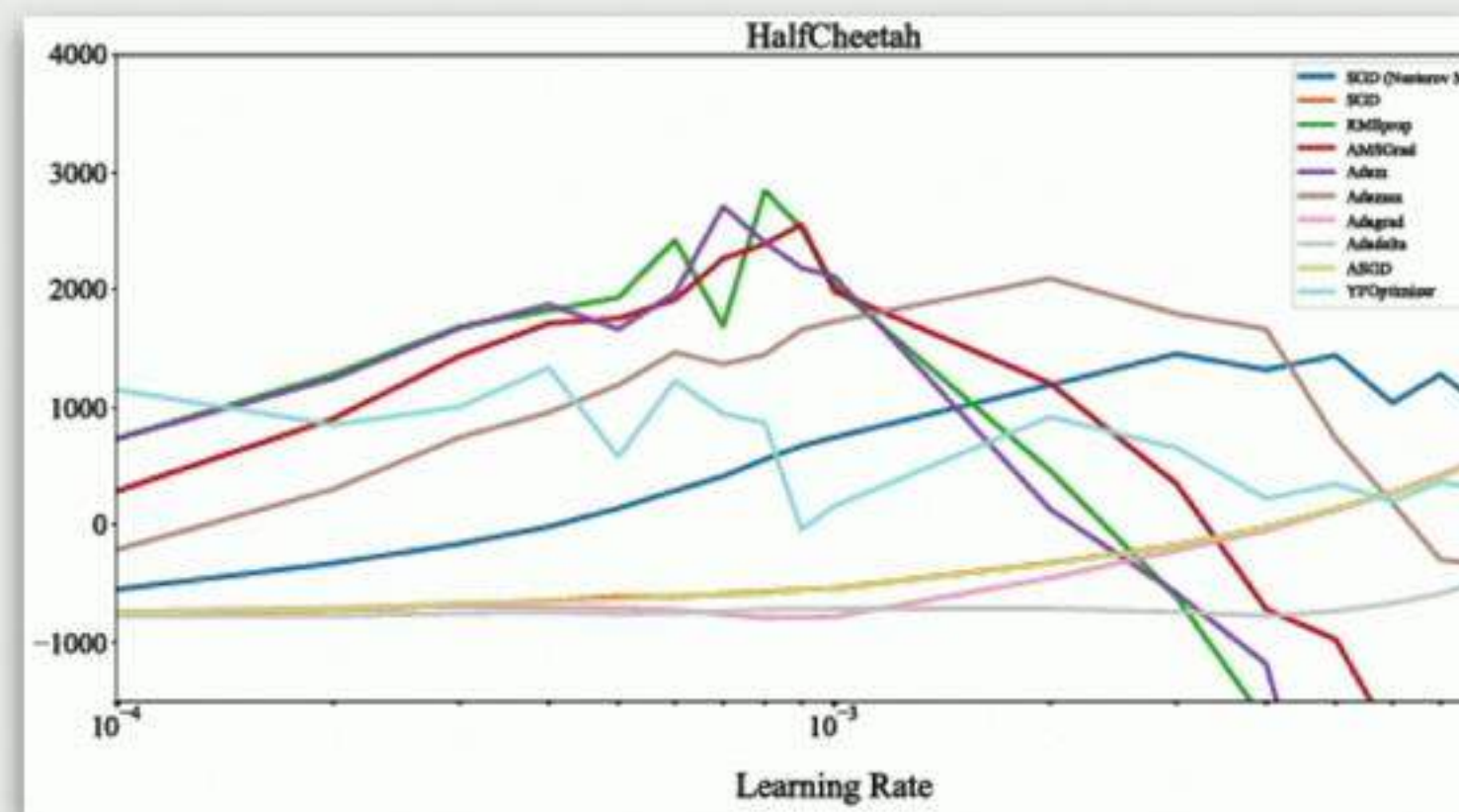


[Henderson et al, 2017a,b] [Lewis et al, 2018]

The Rotten Truth of Deep RL

Deep RL can successfully solve tasks, **but has...**

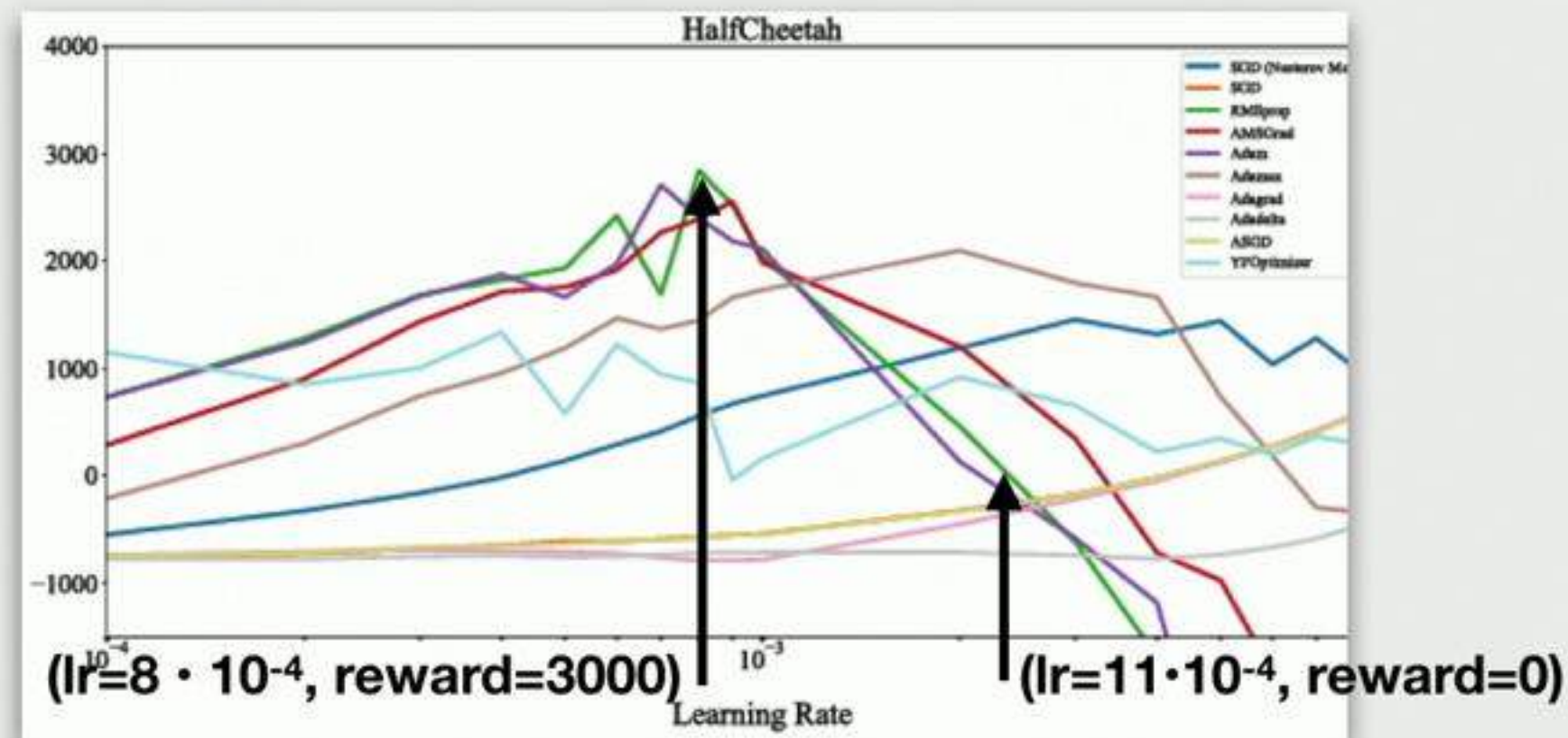
- ▶ Poor reliability over repeated runs
- ▶ High sensitivity to hyperparameters



The Rotten Truth of Deep RL

Deep RL can successfully solve tasks, **but has...**

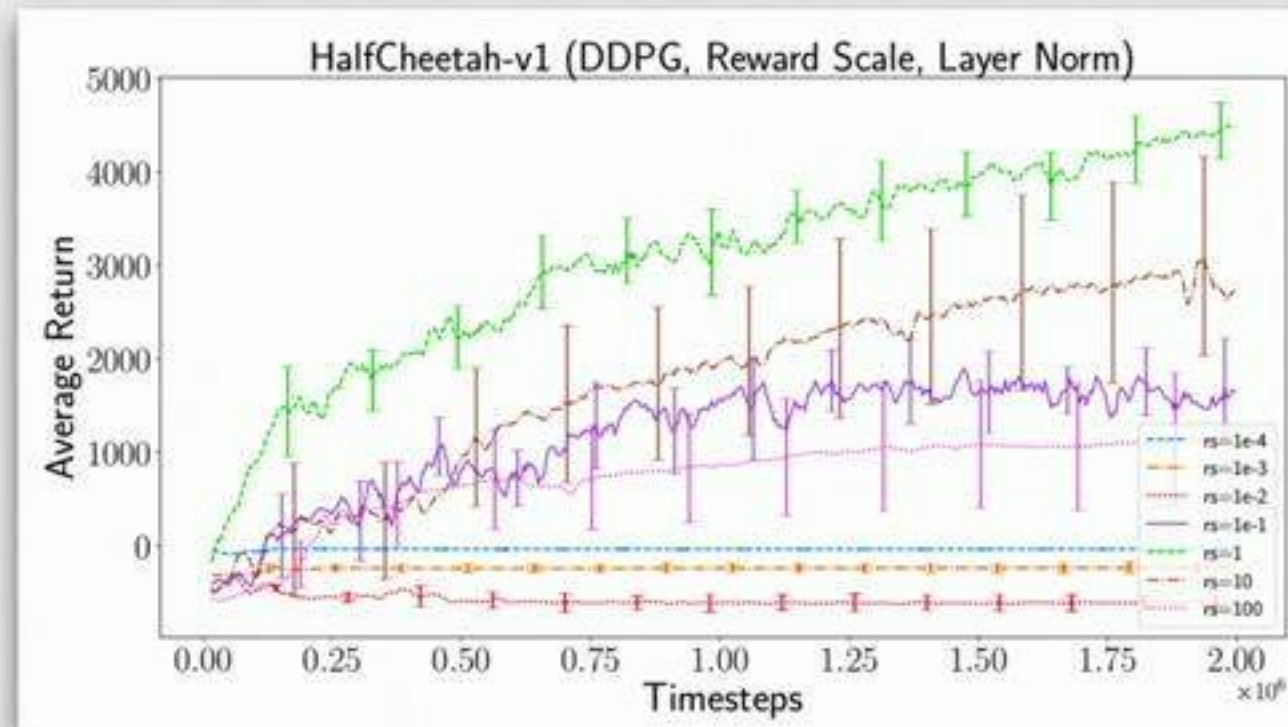
- ▶ Poor reliability over repeated runs
- ▶ High sensitivity to hyperparameters



The Rotten Truth of Deep RL

Deep RL can successfully solve tasks, **but has...**

- ▶ Poor reliability over repeated runs
- ▶ High sensitivity to hyperparameters
- ▶ Poor robustness to environmental artifacts



[Henderson et al, 2017a,b] [Lewis et al, 2018]

The Rotten Truth of Deep RL

Deep RL can successfully solve tasks, **but has...**

- ▶ Poor reliability over repeated runs
- ▶ High sensitivity to hyperparameters
- ▶ Poor robustness to environmental artifacts

Notably, benchmarks don't reveal these problems

The Rotten Truth of Deep RL

Deep RL can successfully solve tasks, **but has...**

- ▶ Poor reliability over repeated runs
- ▶ High sensitivity to hyperparameters
- ▶ Poor robustness to environmental artifacts

Notably, benchmarks don't reveal these problems

Where do such issues come from?

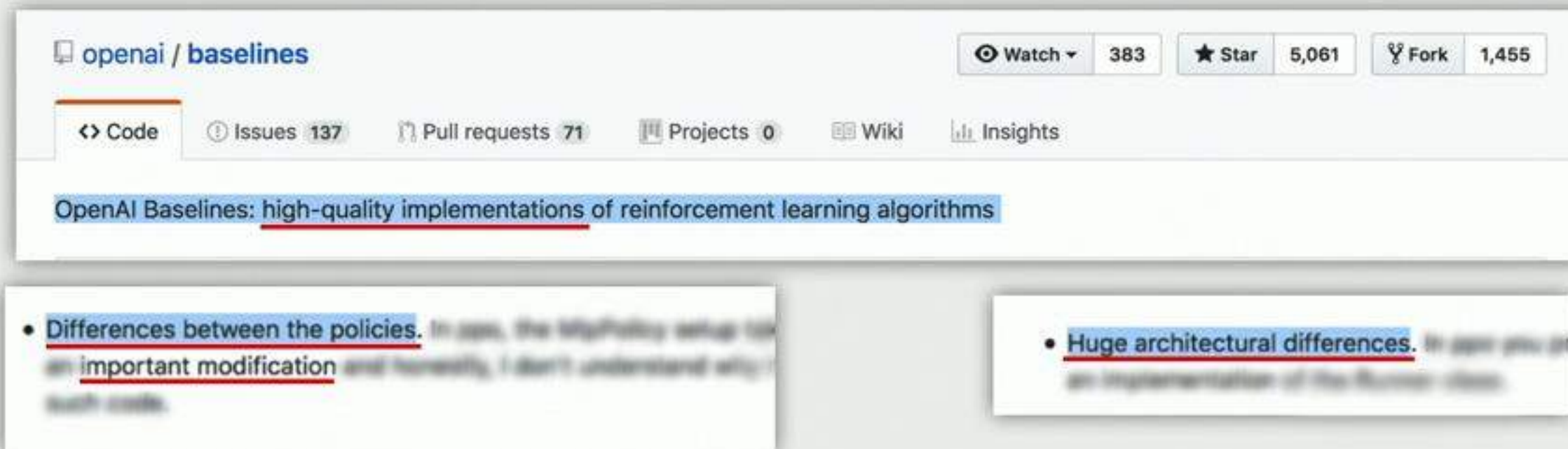
Hard to know: deep RL algorithms have many moving parts!

Implementation Obscures Deep RL Algorithms



Source: GitHub issues

Implementation Obscures Deep RL Algorithms



The screenshot shows the GitHub repository page for 'openai / baselines'. The repository has 383 watchers, 5,061 stars, and 1,455 forks. The 'Code' tab is selected, showing the repository description: 'OpenAI Baselines: high-quality implementations of reinforcement learning algorithms'. Below the description, two GitHub issues are visible. The first issue on the left is titled 'Differences between the policies.' and mentions 'an important modification'. The second issue on the right is titled 'Huge architectural differences.'.

openai / baselines

Watch 383 Star 5,061 Fork 1,455

Code Issues 137 Pull requests 71 Projects 0 Wiki Insights

OpenAI Baselines: high-quality implementations of reinforcement learning algorithms

- Differences between the policies. In appn, the highPolicy setup has an important modification and honestly, I don't understand why it's not in the code.
- Huge architectural differences. In appn you get an implementation of the thinner class.

Source: GitHub issues

Implementation Obscures Deep RL Algorithms

openai / baselines

Watch 383 Star 5,061 Fork 1,455

Code Issues 137 Pull requests 71 Projects 0 Wiki Insights

OpenAI Baselines: high-quality implementations of reinforcement learning algorithms

- Differences between the policies. In `ppo`, the `klPolicy` setup has an important modification and honestly, I don't understand why it's there.
- Huge architectural differences. In `ppo` you get an implementation of the `ActorCritic` class.
- Nontrivial changes to the paper, part 2. The code is sprinkled with small tricks. For `ppo` you apply a normalization to the advantage function, `norm`. This one could be my fault. But I've only seen something like this in `dueling` or `learning`. You subtract the mean.
- Nontrivial changes to the paper. I read the `openai` blogpost and the `ppo` paper. In `ppo` you use a clipped ratio of action probabilities. `ppo` does that. `openai` also uses the value function. I would consider that a big difference between `ppo` and `ppo`.

Source: GitHub issues

Implementation Obscures Deep RL Algorithms

openai / baselines

Watch

383

Star

5,061

Fork

1,455

Deep RL algorithms are complicated &
underspecified!

an important modification and honestly, I don't understand why
both code.

an implementation of the PPO paper

- Nontrivial changes to the paper, part 2. The code is sprinkled with small tricks. For
example a normalization to the advantage function, here. This one could be my last
but I've only seen something like this in Dueling a learning. You subtract the mean

epthadow commented 24 days ago

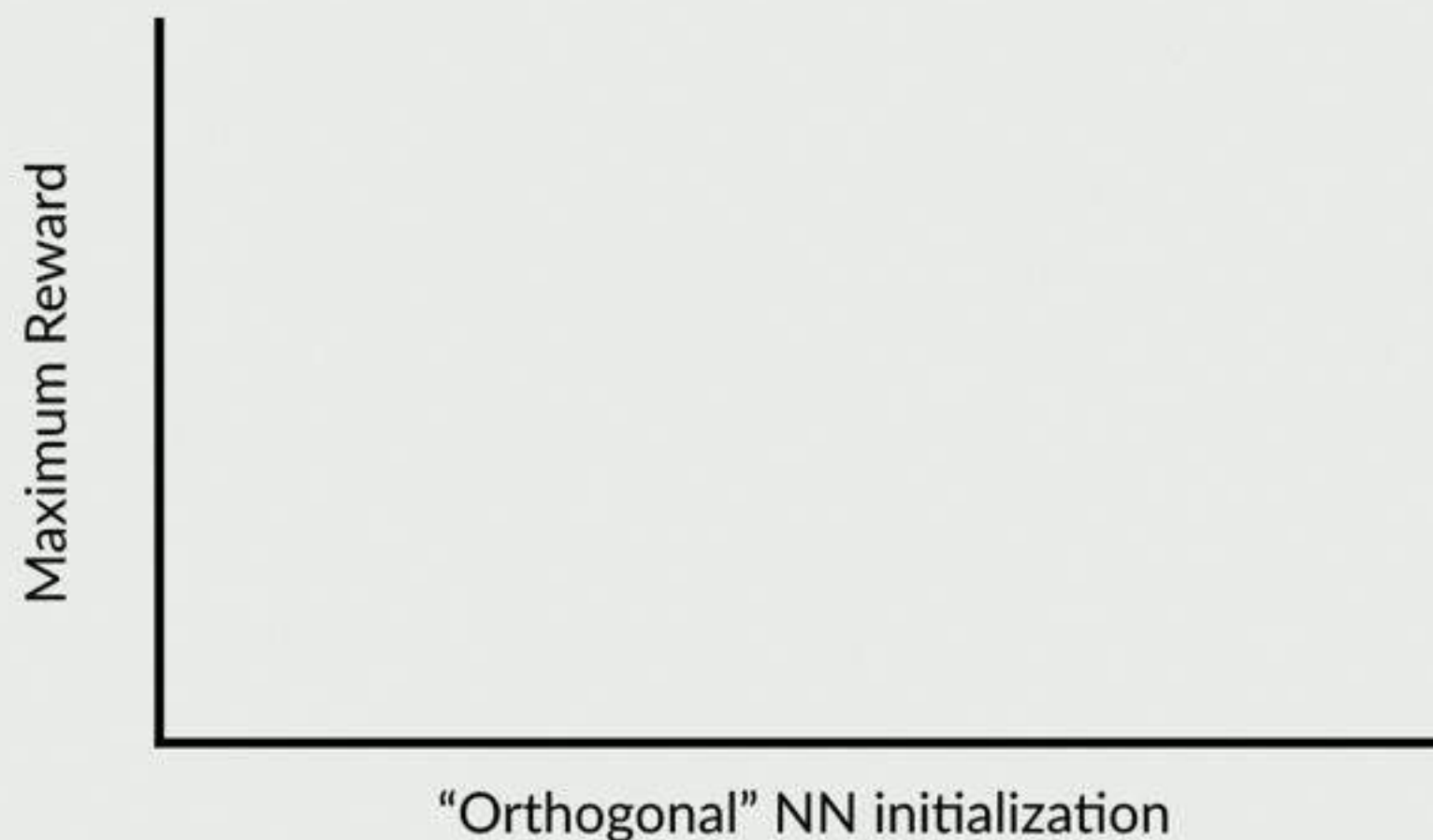
There is one thing between PPO1 and PPO2 that I don't understand. My
the model actually contains both the old and new set of parameters, a

Nontrivial changes to the paper. I read the openai blogpost and the ppo paper.
I'm looking at a clipped ratio of action probabilities. openai says that ppo1 also use
the value function. I would consider that a big difference between ppo and ppo

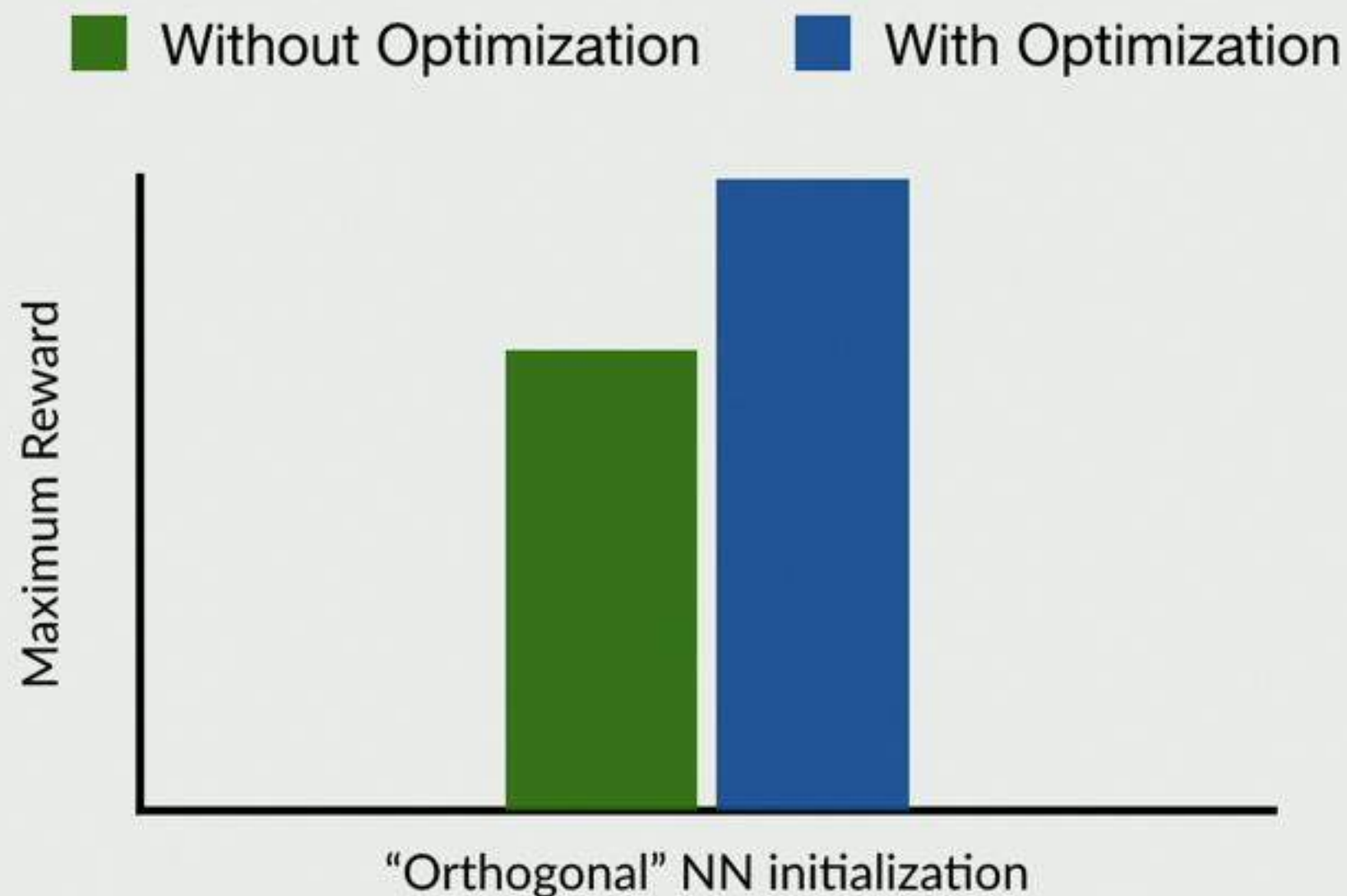
Source: GitHub issues

Implementation Obscures Deep RL Algorithms

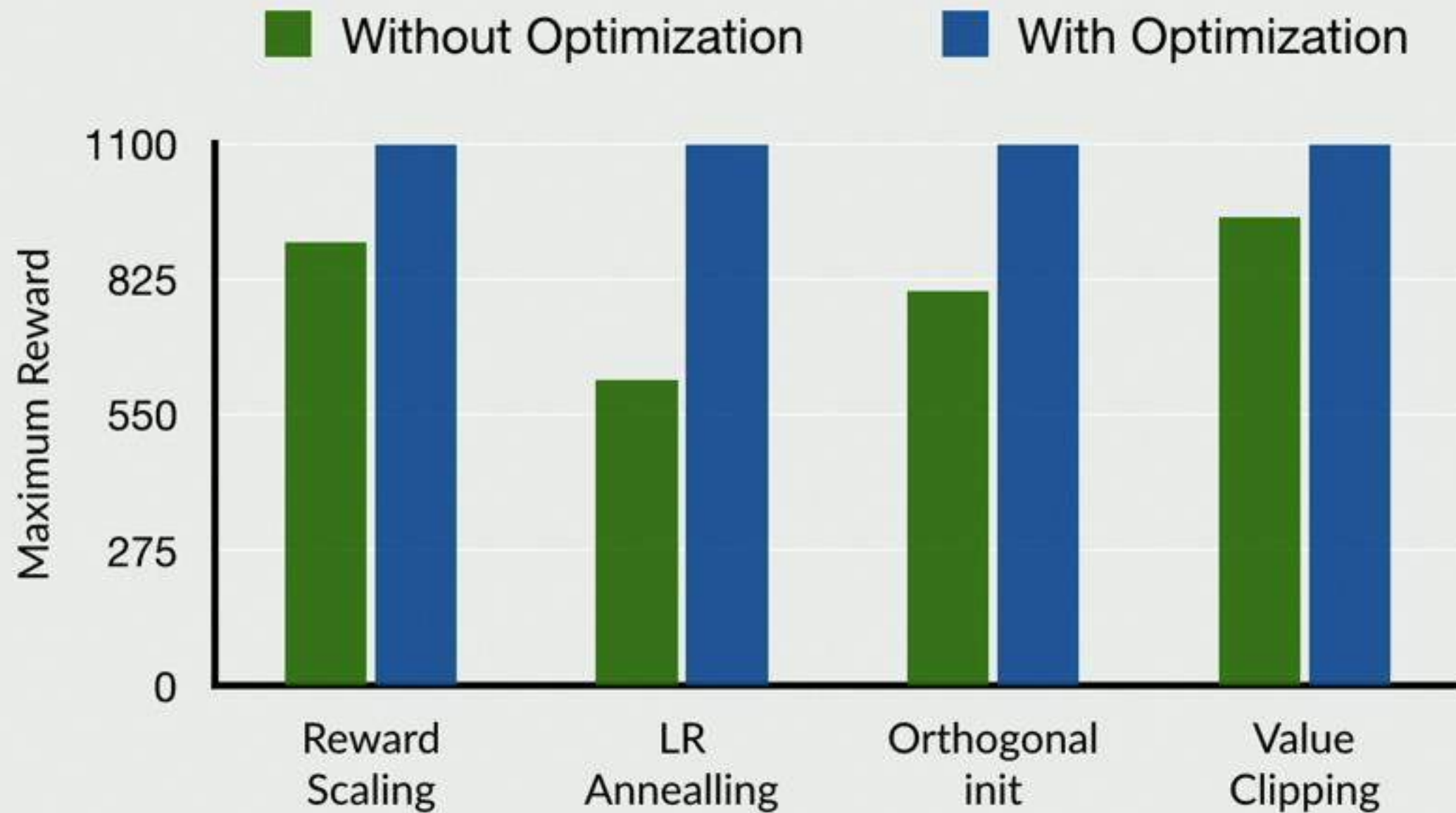
■ Without Optimization ■ With Optimization



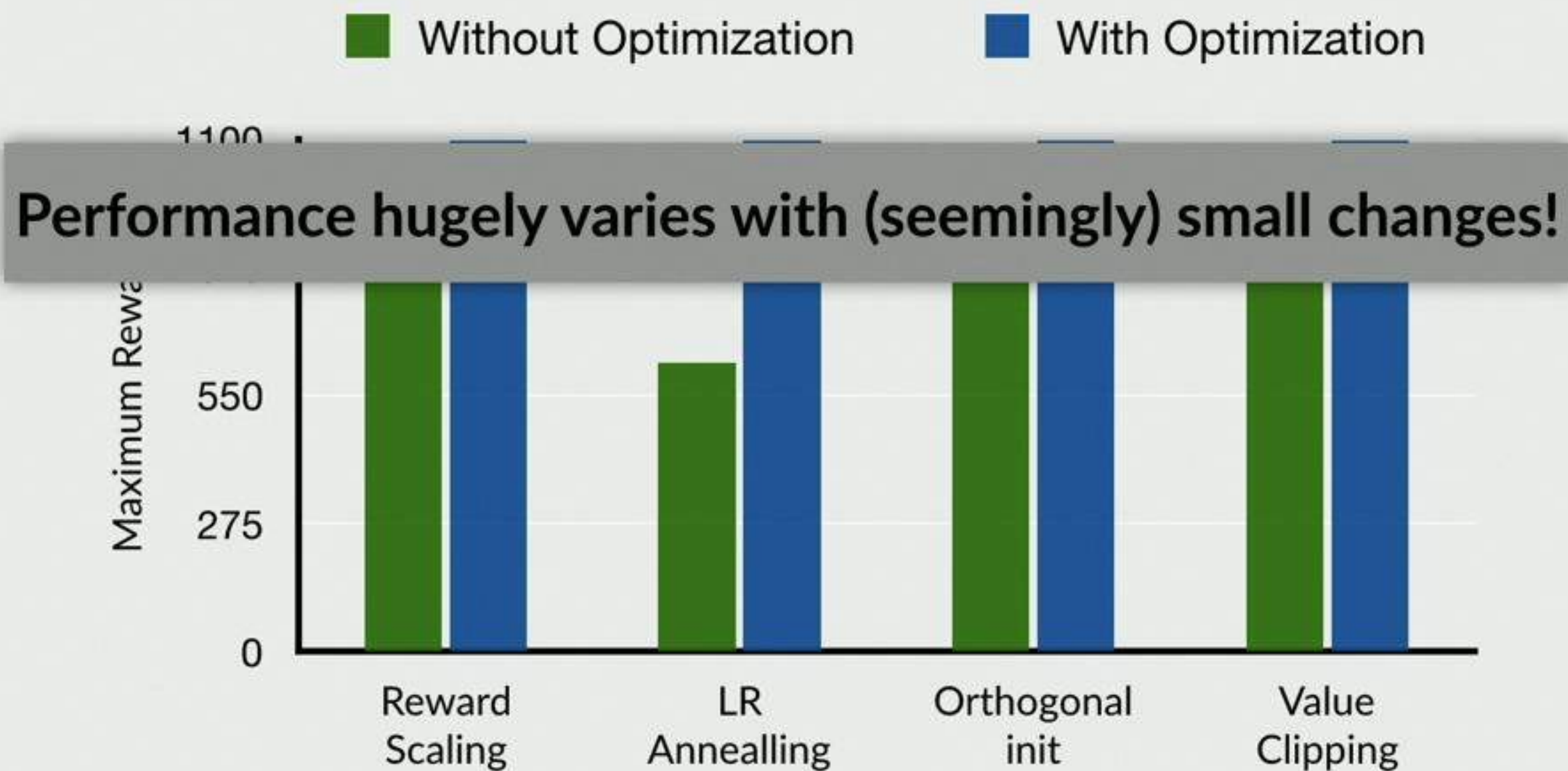
Implementation Obscures Deep RL Algorithms



Implementation Obscures Deep RL Algorithms



Implementation Obscures Deep RL Algorithms



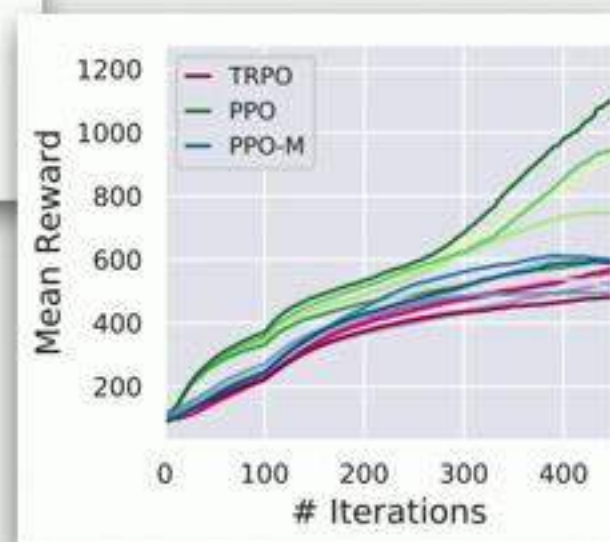
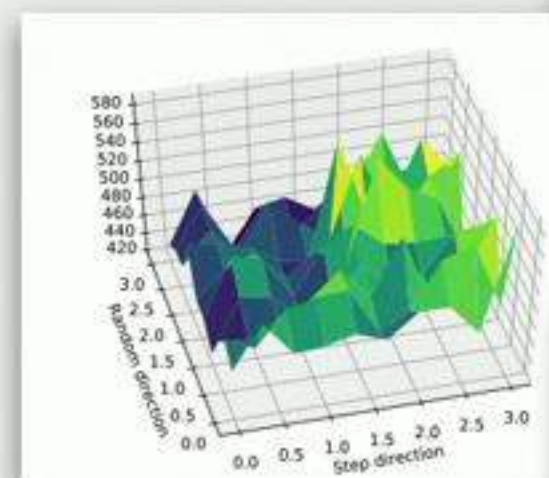
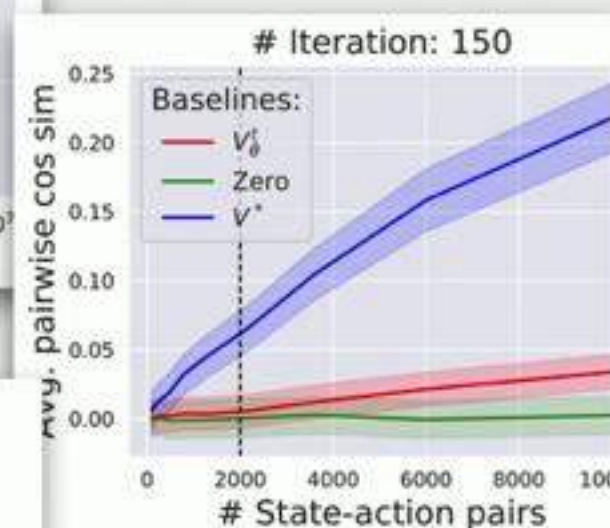
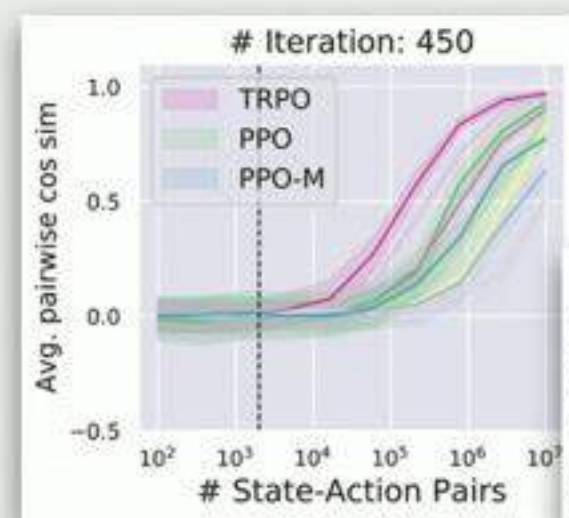
Implementation Obscures Deep RL Algorithms

- ▶ Deep RL methods are complicated & underspecified
- ▶ Reasons for unreliability, performance are unclear
- ▶ Deep RL methods are poorly understood!

Back to First Principles

Back to First Principles

- ▶ Gradient Estimates
- ▶ Value Prediction
- ▶ Optimization Landscapes
- ▶ Trust Regions



Gradient Estimation

Key assumption of policy gradient framework:

$$\nabla_{\theta} \mathbb{E}_{\tau \sim \theta}[R(\theta)] \approx \frac{1}{N} \sum_{\tau \sim \theta} g(\tau)$$

Gradient Estimation

Key assumption of policy gradient framework:

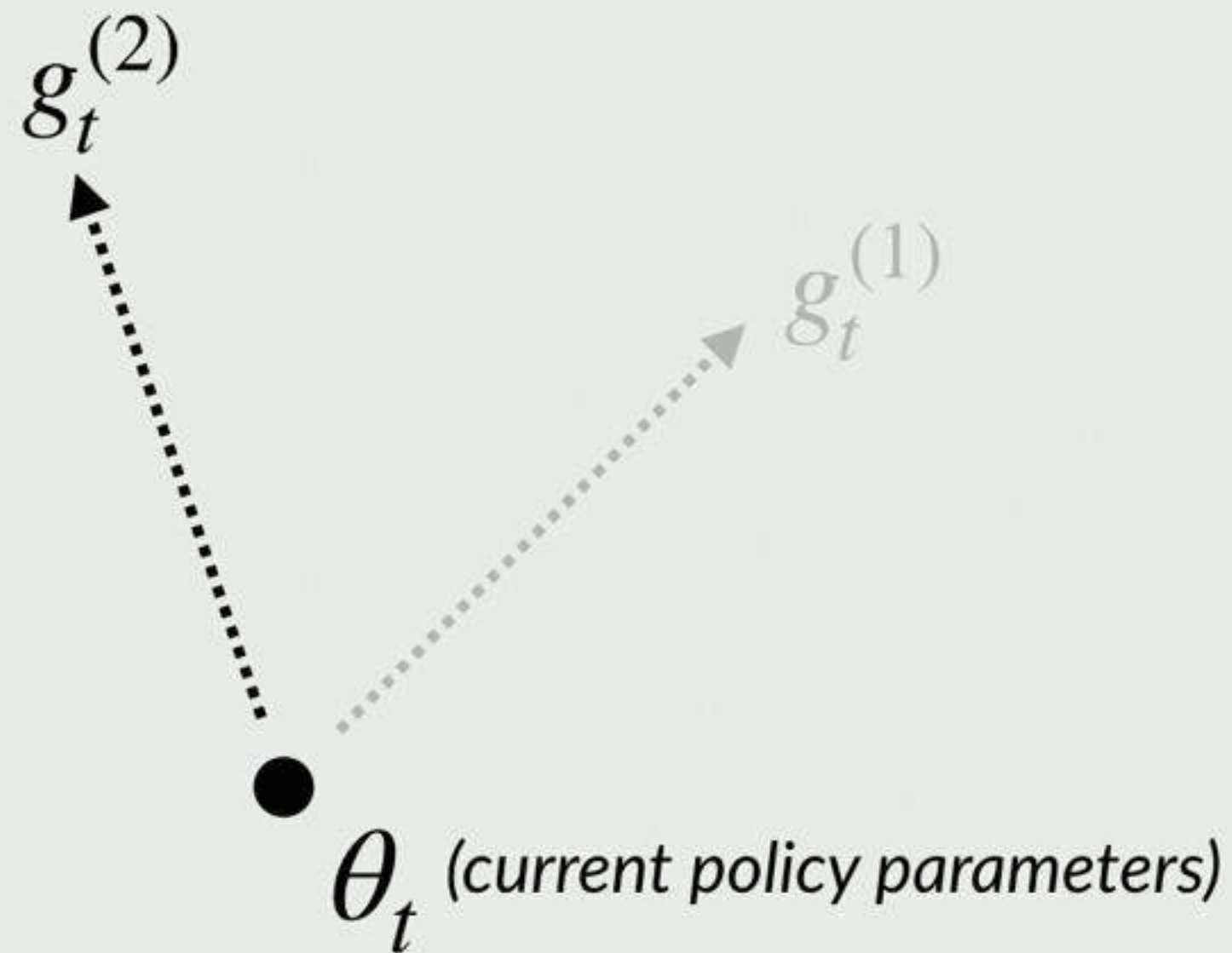
$$\nabla_{\theta} \mathbb{E}_{\tau \sim \theta}[R(\theta)] \approx \frac{1}{N} \sum_{\tau \sim \theta} g(\tau)$$

How valid is this?

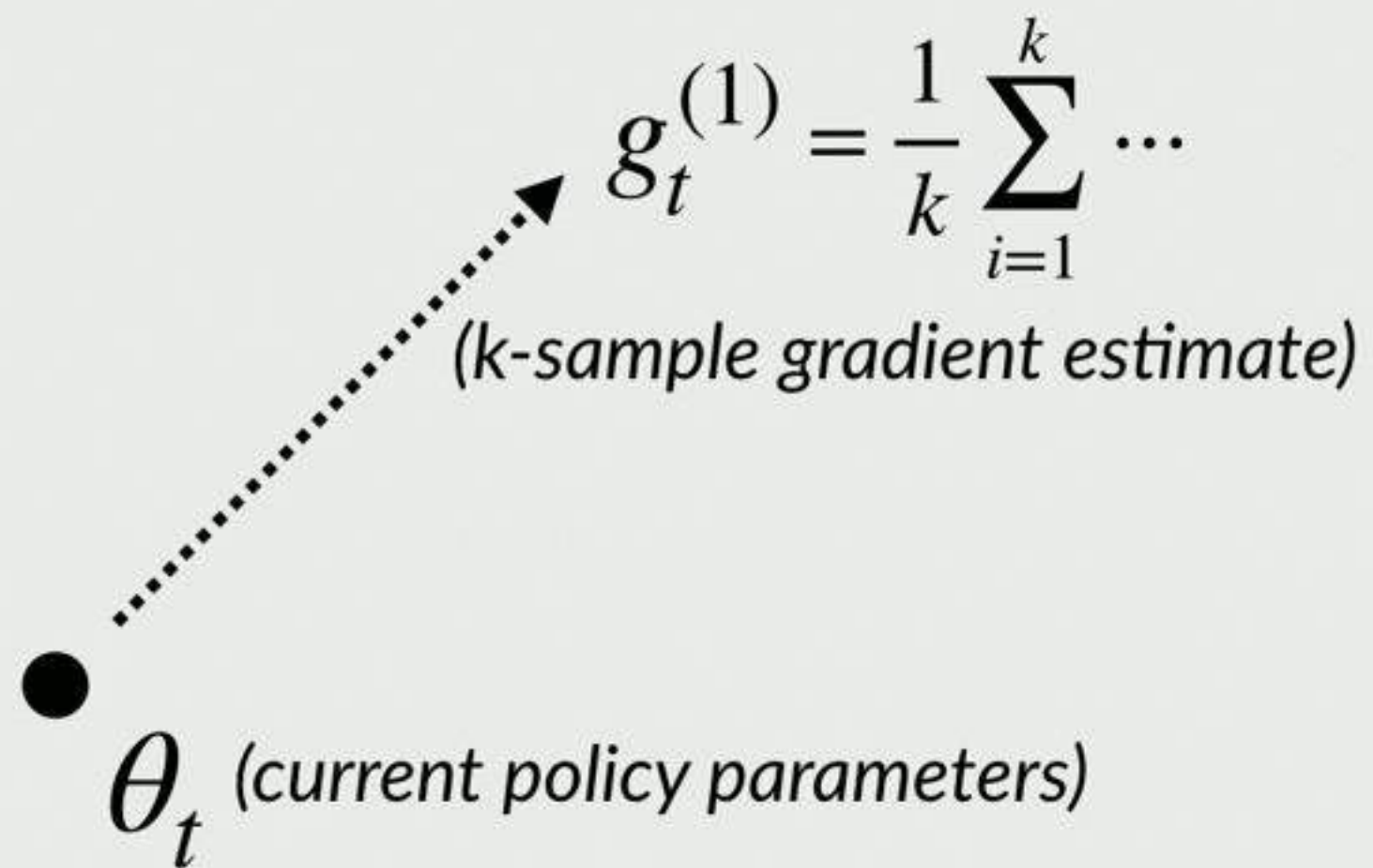
Gradient Estimation

- θ_t (current policy parameters)

Gradient Estimation



Gradient Estimation



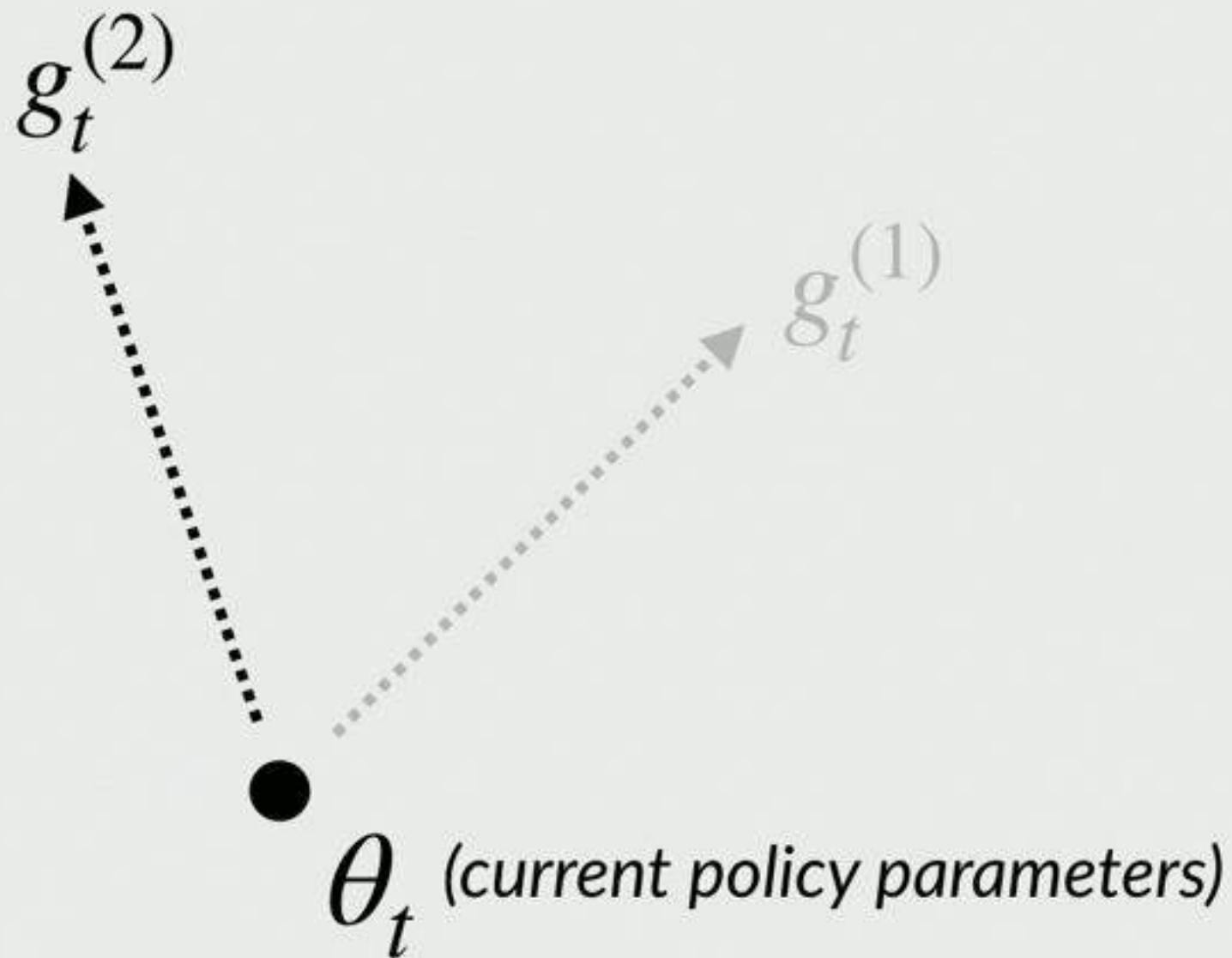
A diagram illustrating gradient estimation. A solid black dot is located in the lower-left quadrant. A dotted arrow originates from this dot and points diagonally upwards and to the right, ending at the first part of a mathematical formula. The formula is $g_t^{(1)} = \frac{1}{k} \sum_{i=1}^k \dots$. Below the arrow, the text "(k-sample gradient estimate)" is written. Below the dot, the text " θ_t (current policy parameters)" is written.

$$g_t^{(1)} = \frac{1}{k} \sum_{i=1}^k \dots$$

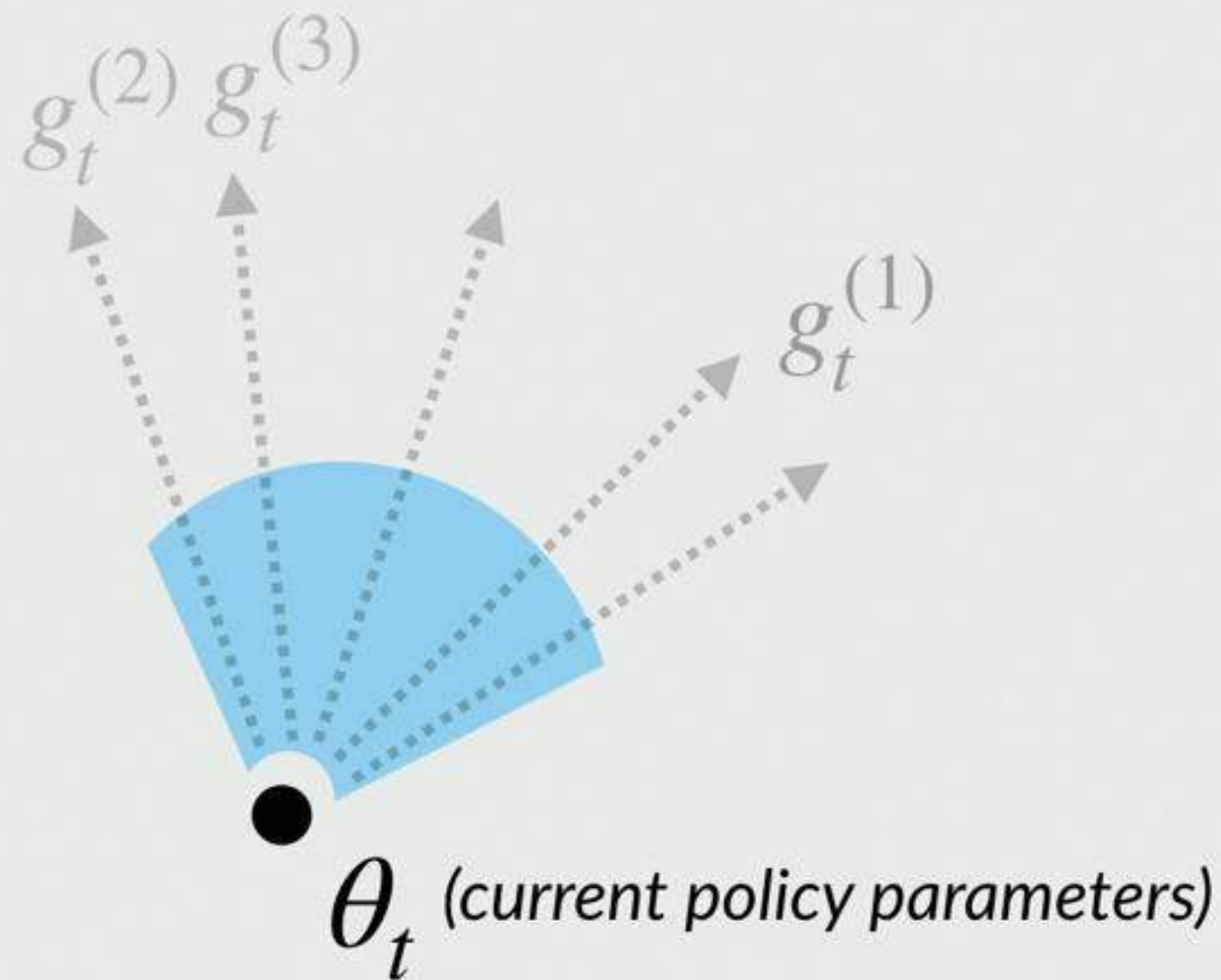
(k-sample gradient estimate)

θ_t (current policy parameters)

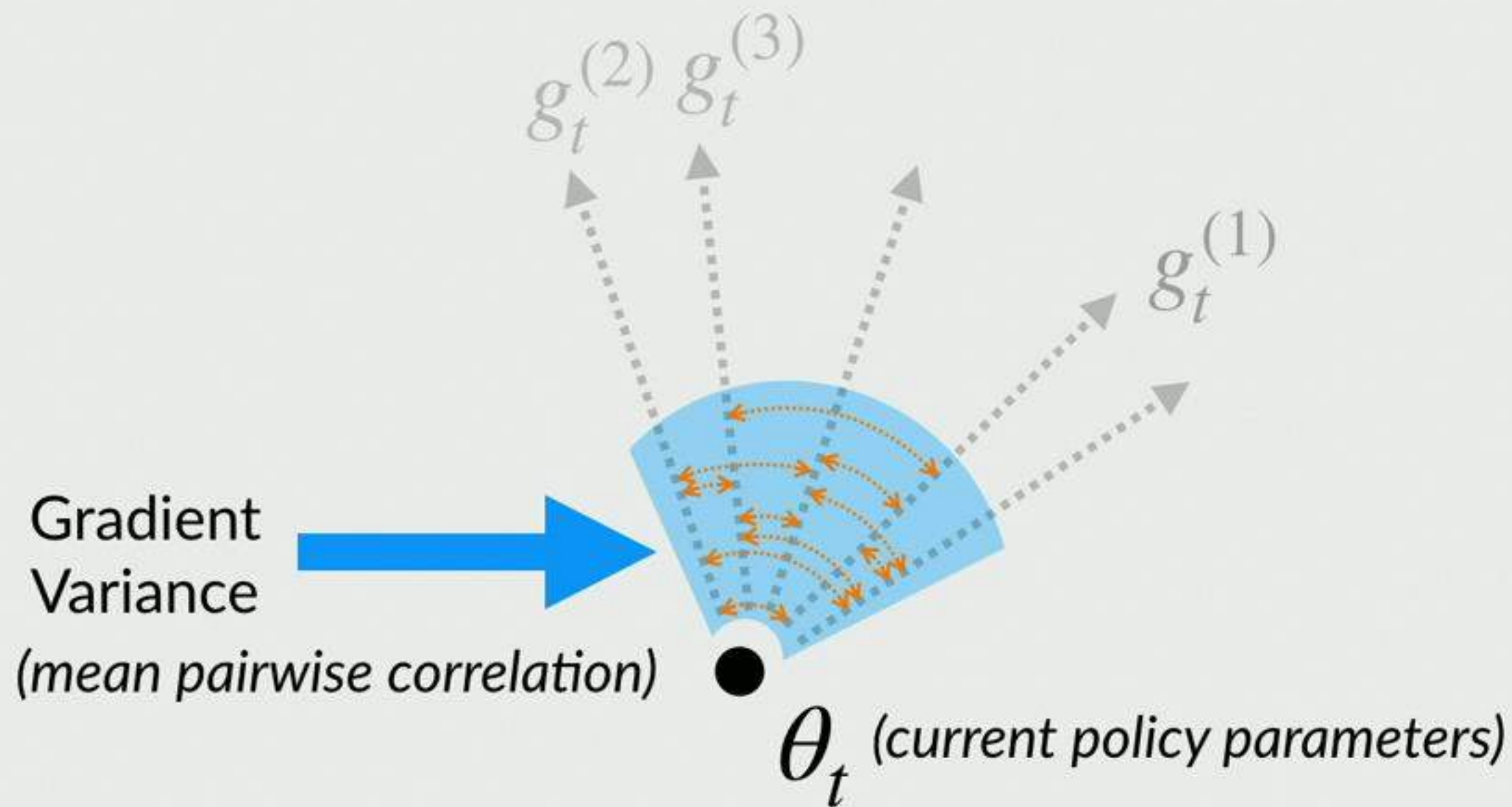
Gradient Estimation



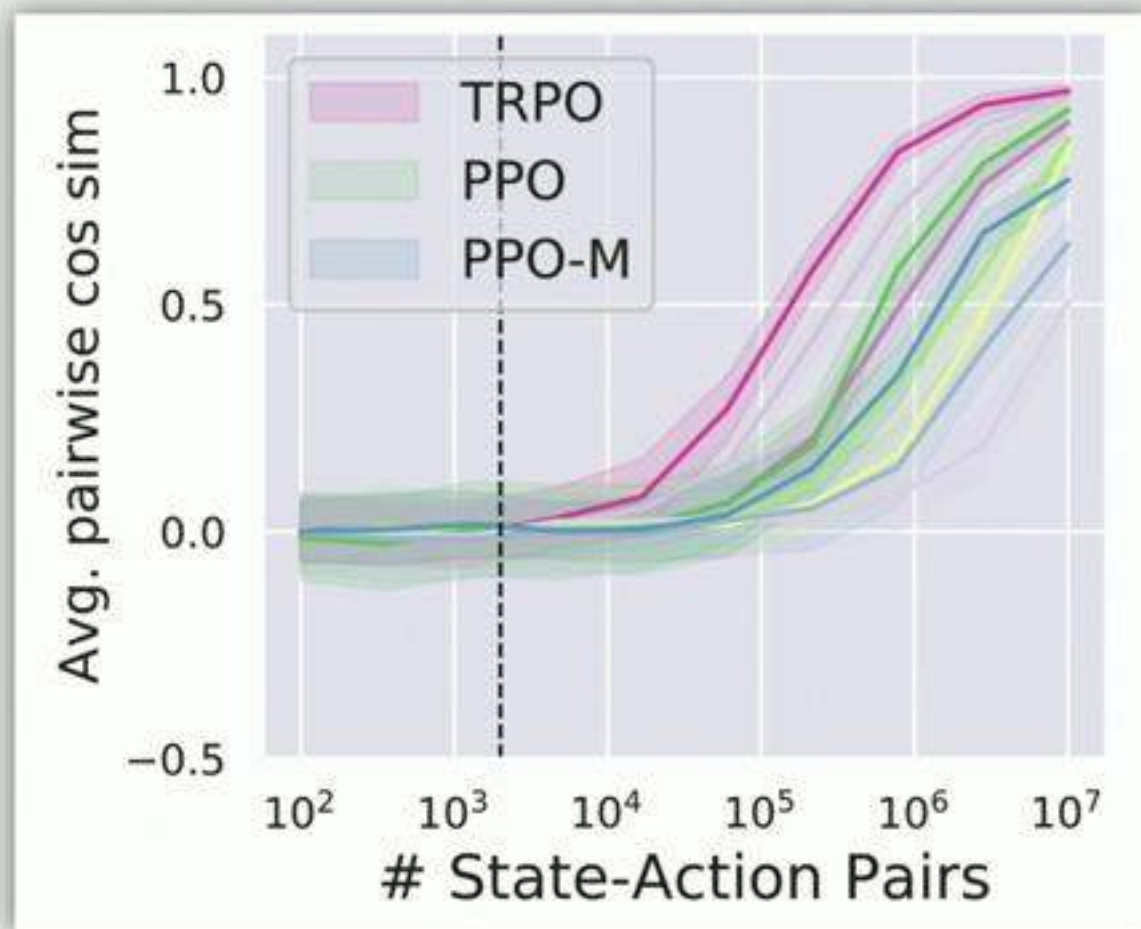
Gradient Estimation



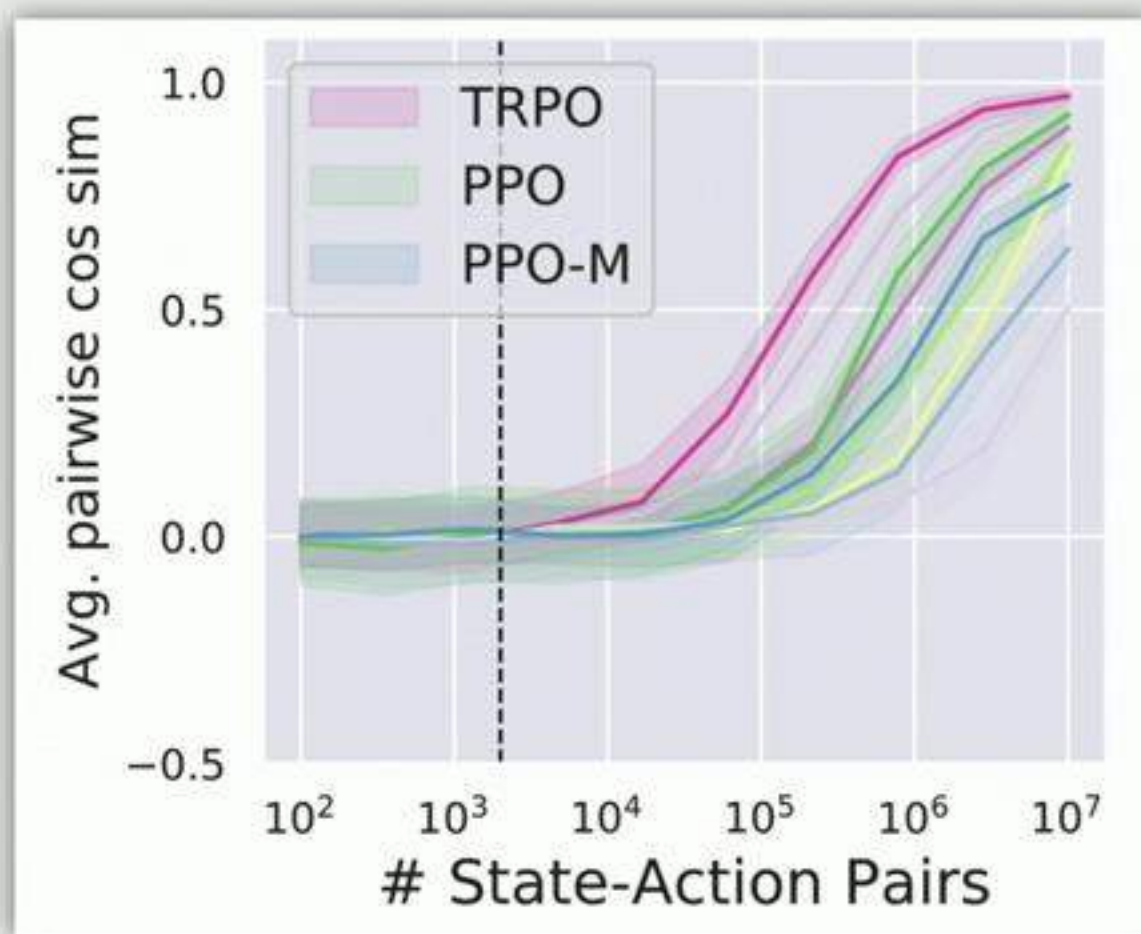
Gradient Estimation



Gradient Variance



Gradient Variance

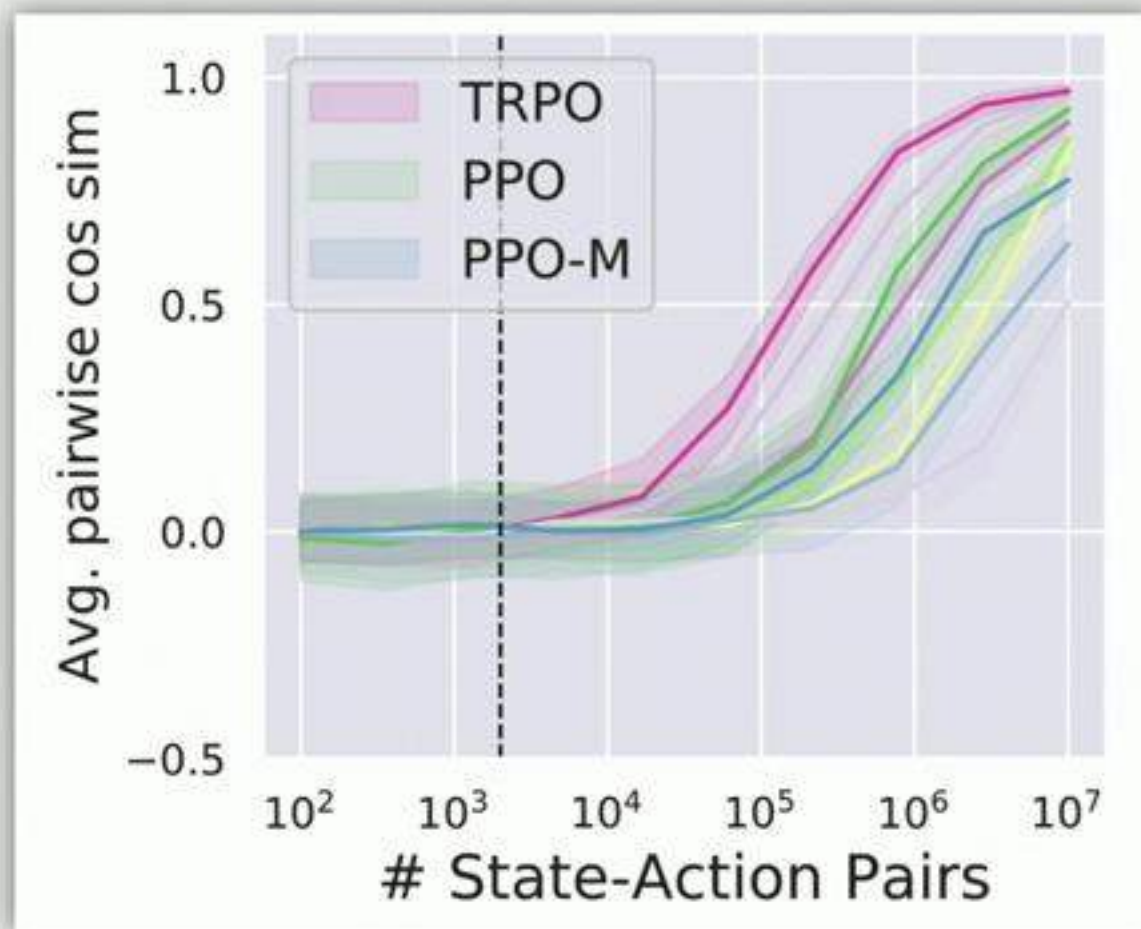


- ▶ **Black line:** relevant sample regime
- ▶ Gradients are less concentrated than they could be
- ▶ Less correlated for “harder” tasks, later iterations

Gradient Estimation

- ▶ No good understanding of training dynamics
 - ▶ How does variance influence optimization?
 - ▶ Can we use insights from stochastic opt?
- ▶ Missing a link from reliability to sample size

Gradient Variance



Gradient Estimation

- ▶ No good understanding of training dynamics
 - ▶ How does variance influence optimization?
 - ▶ Can we use insights from stochastic opt?
- ▶ Missing a link from reliability to sample size

Value Prediction

Gradient estimation is hindered by high variance!

Observation: If we can estimate the **value** of a state, can significantly lower variance

(The **value of a state** is the cumulative expected reward received after visiting the state)

Intuition: Need to separate action quality from state quality

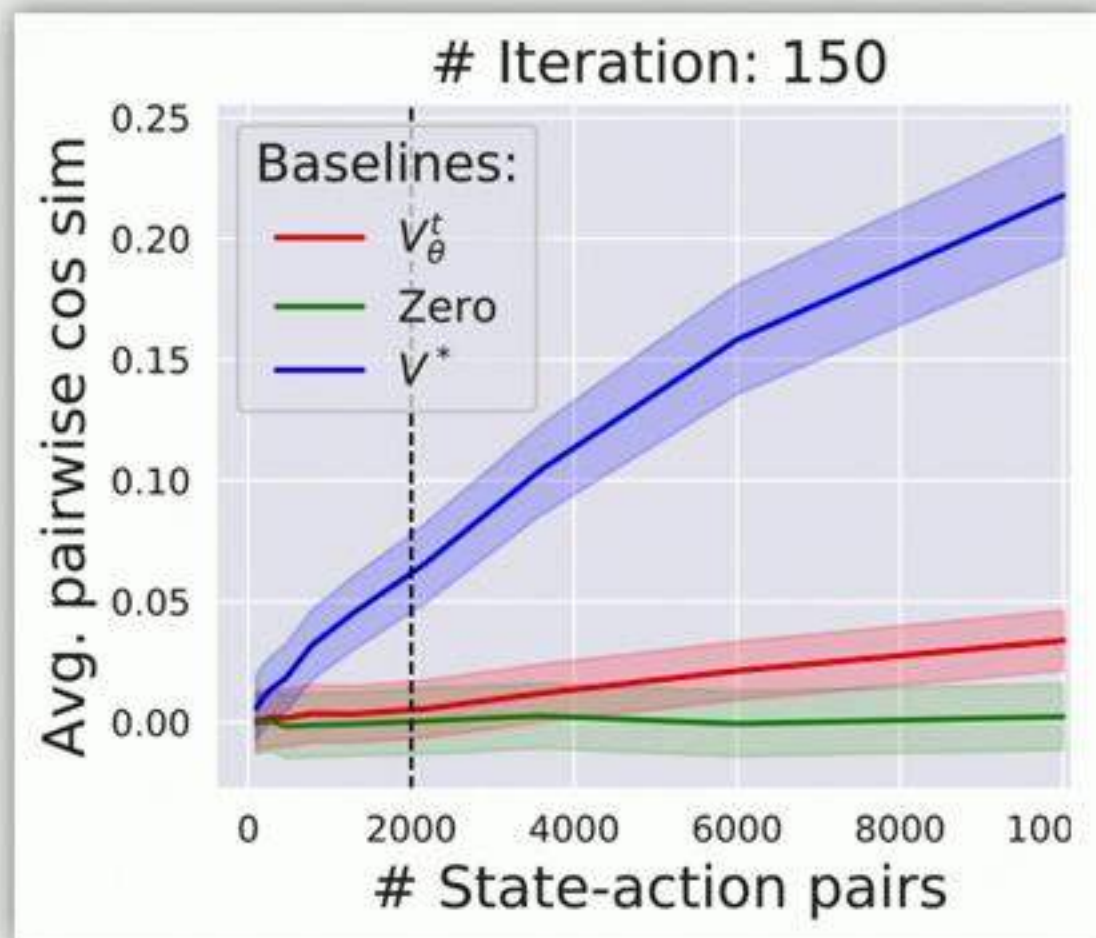
Value Prediction

Variance reduction needs **good value estimates**

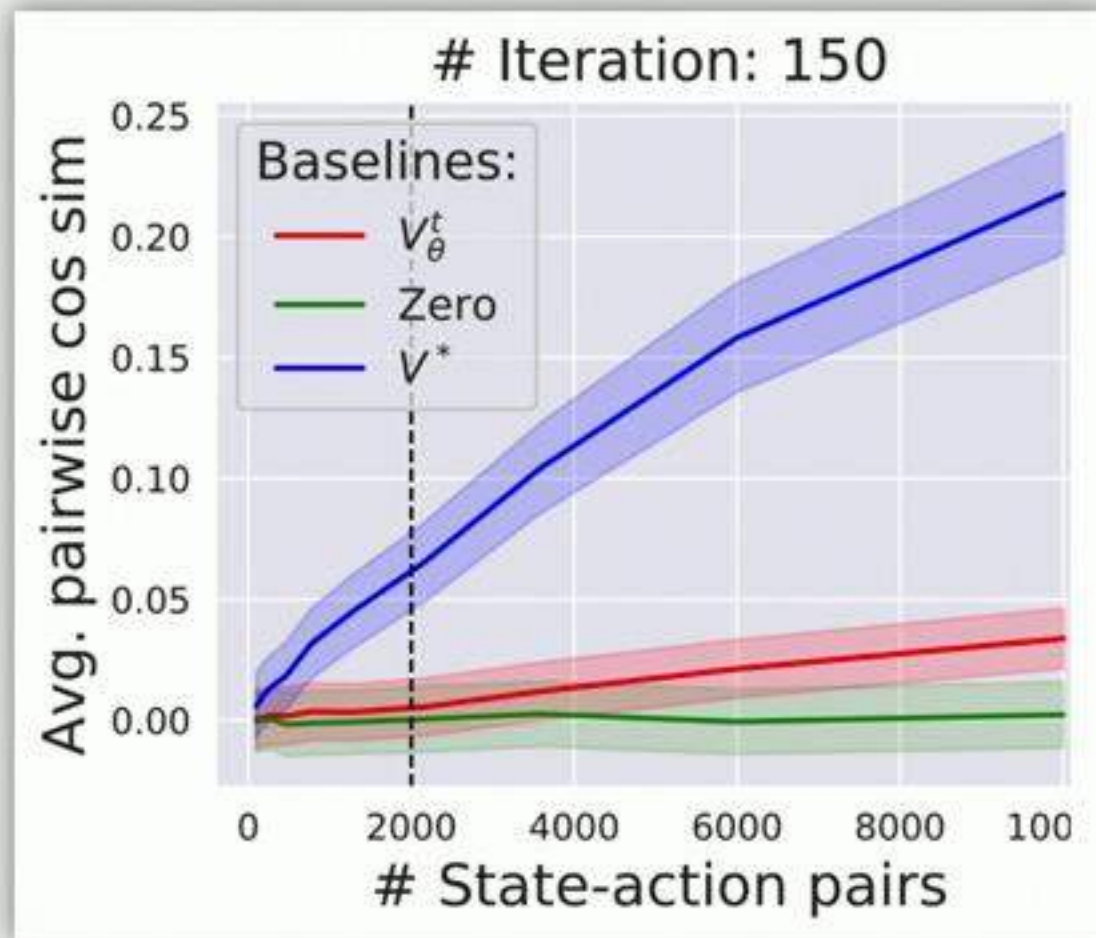
In Deep RL, values come from a neural network

To what degree do we actually reduce variance?

Value Prediction



Value Prediction



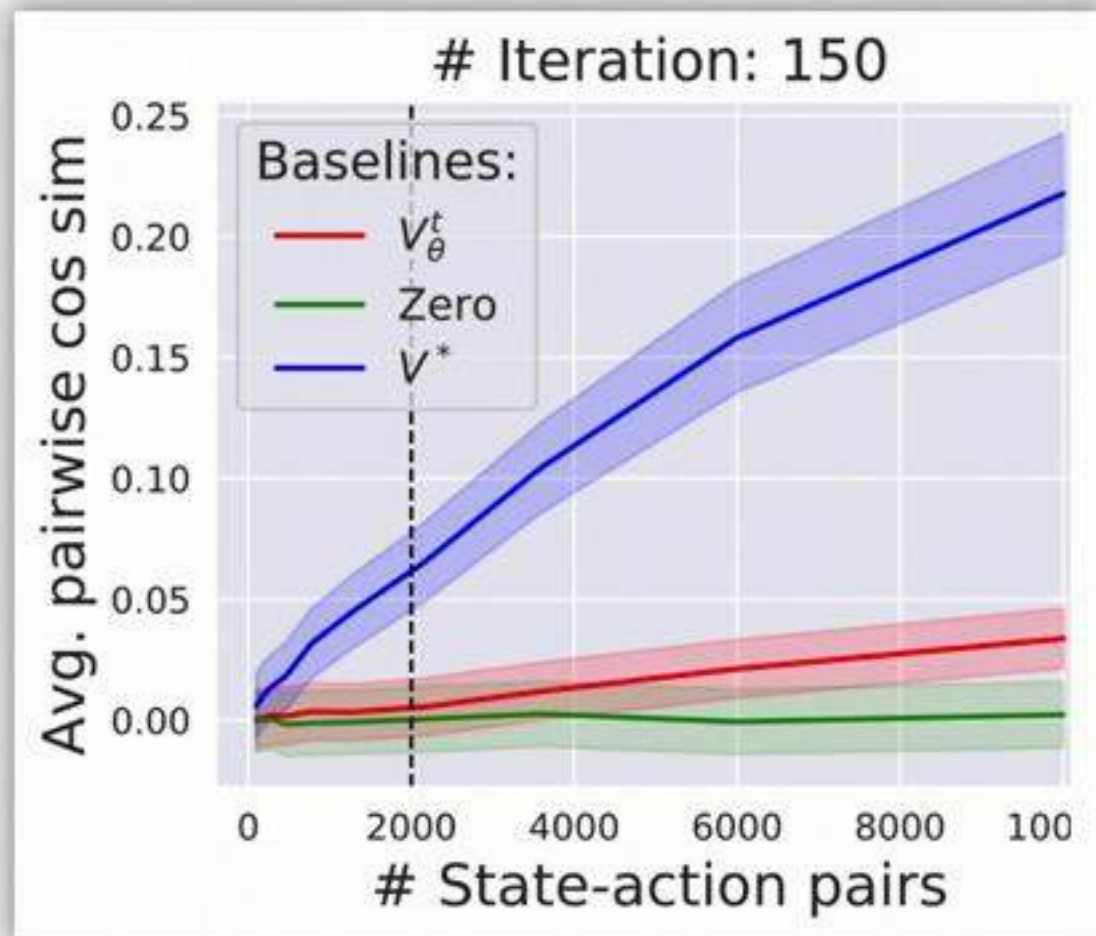
True value function

Agent's value function

No value function

Agent does significantly worse than optimal!

Value Prediction



True value function

Agent's value function

No value function

- ▶ Might **look** small, but using a value network makes **big** difference
- ▶ How would using the true value affect training?
- ▶ Can we get better value estimates?

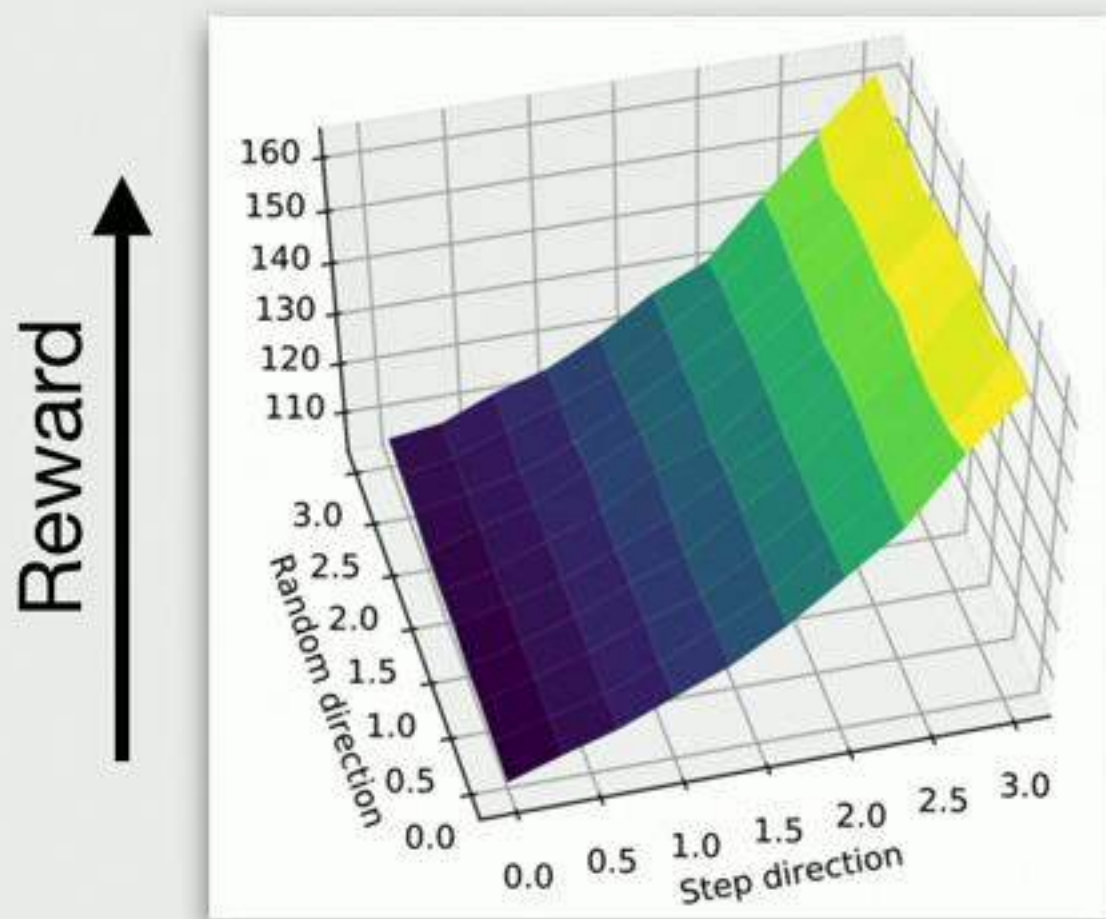
Optimization Landscapes

Assumption: taking gradient steps increases reward

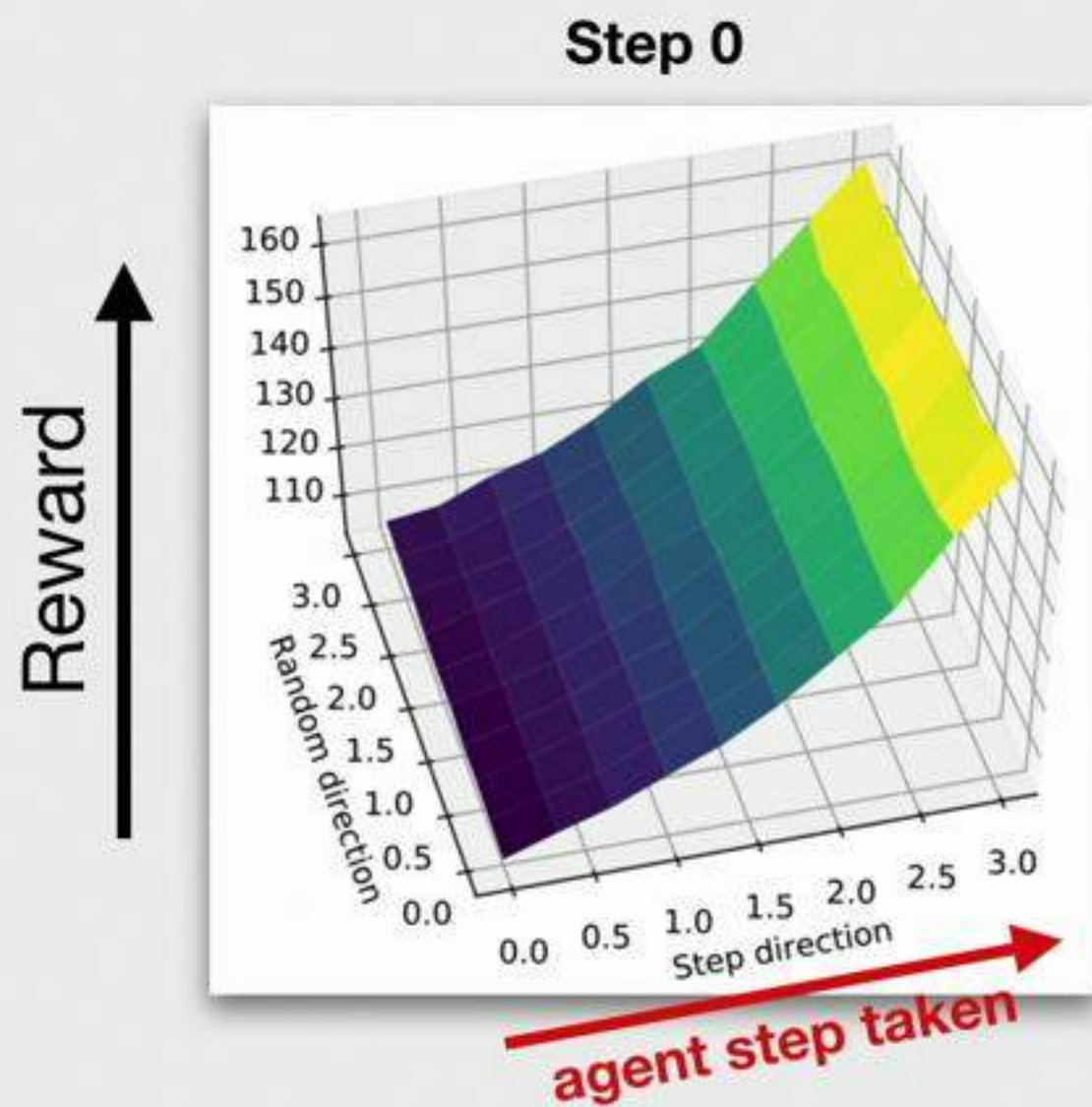
How valid is this assumption in practice?

Optimization Landscapes

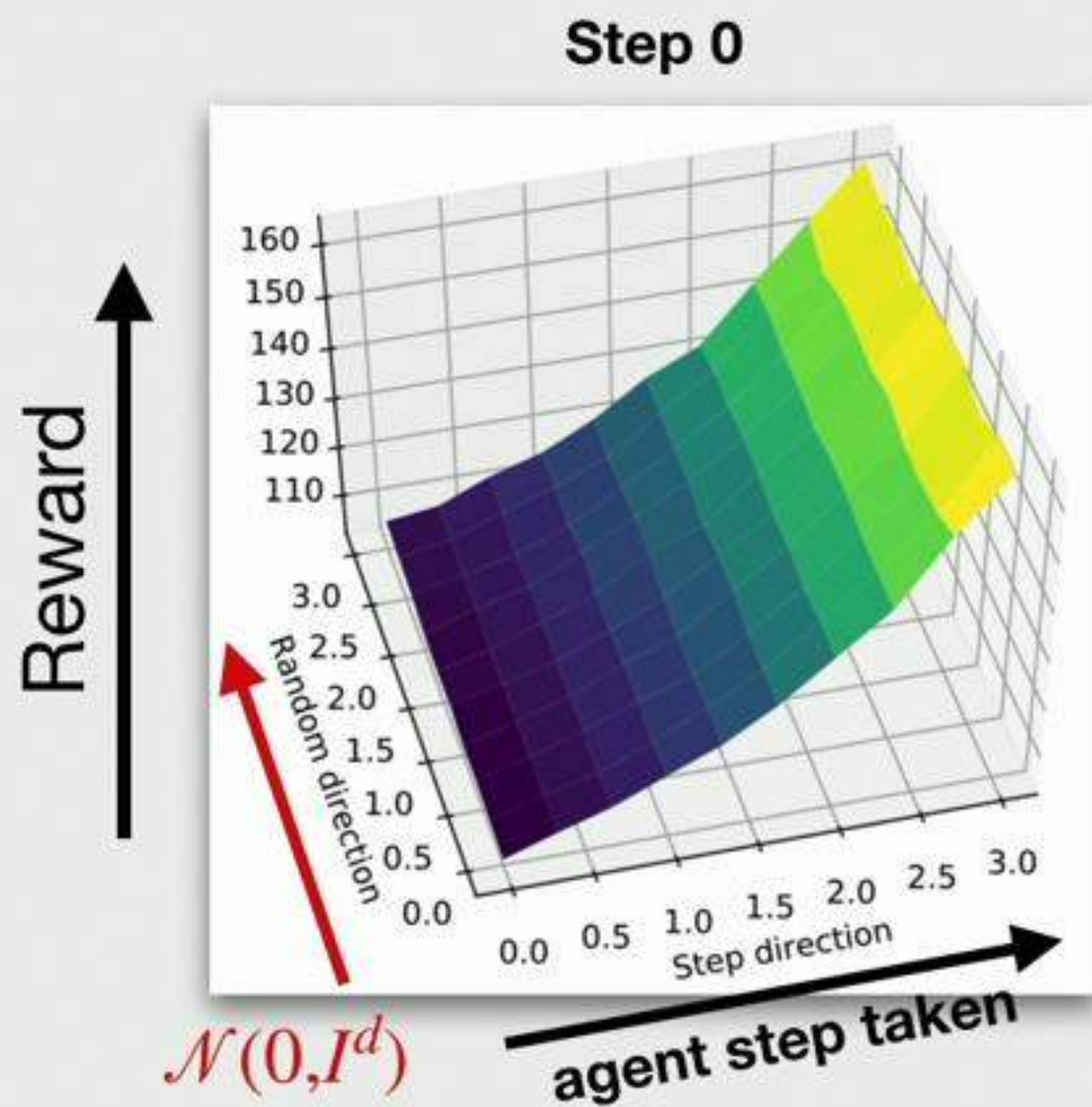
Step 0



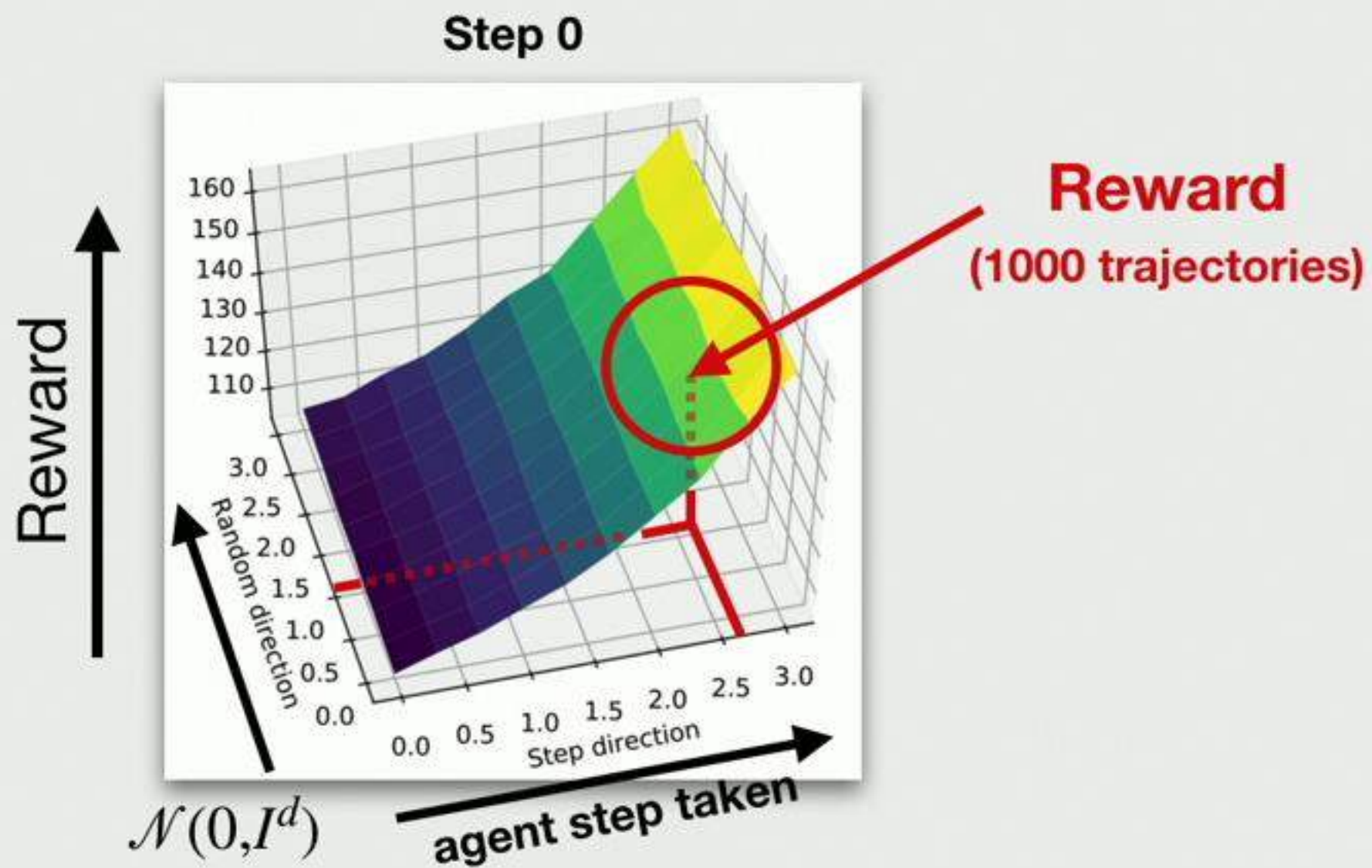
Optimization Landscapes



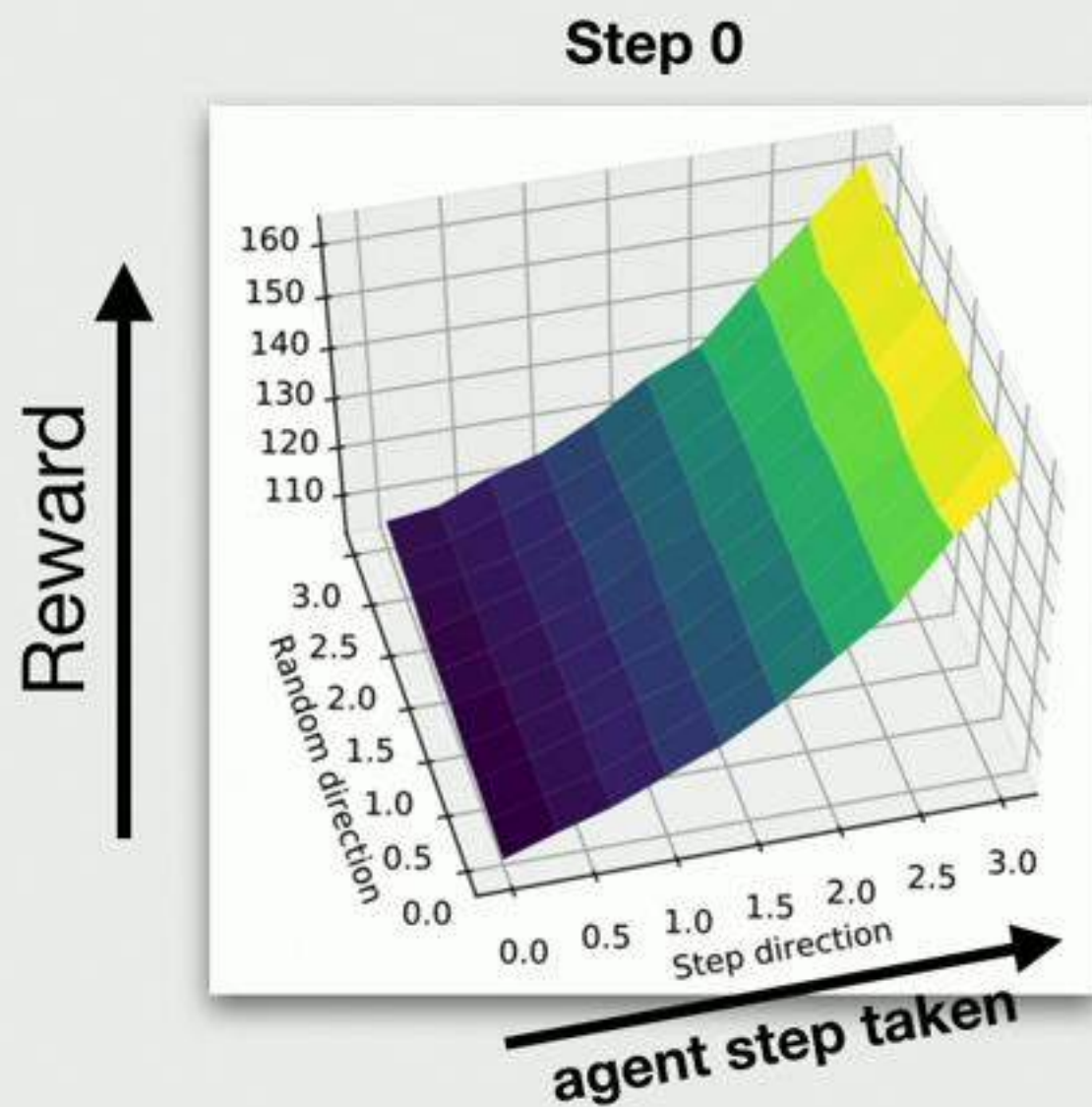
Optimization Landscapes



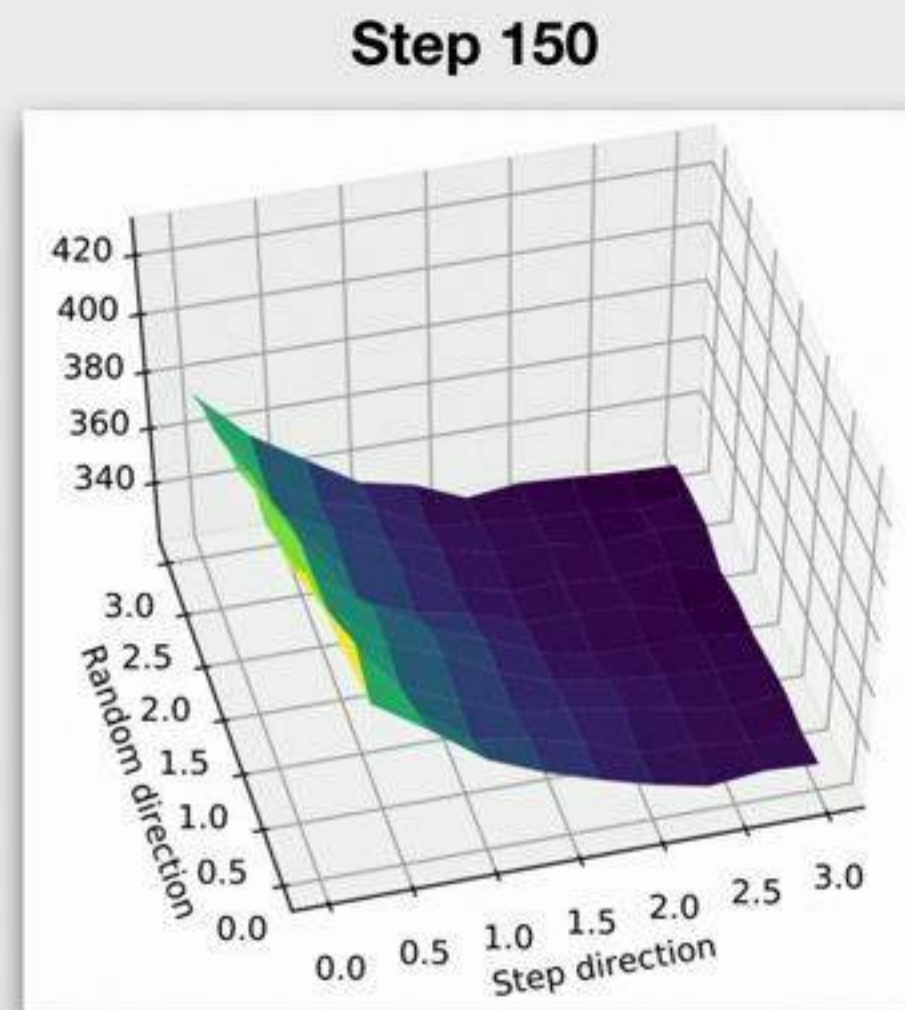
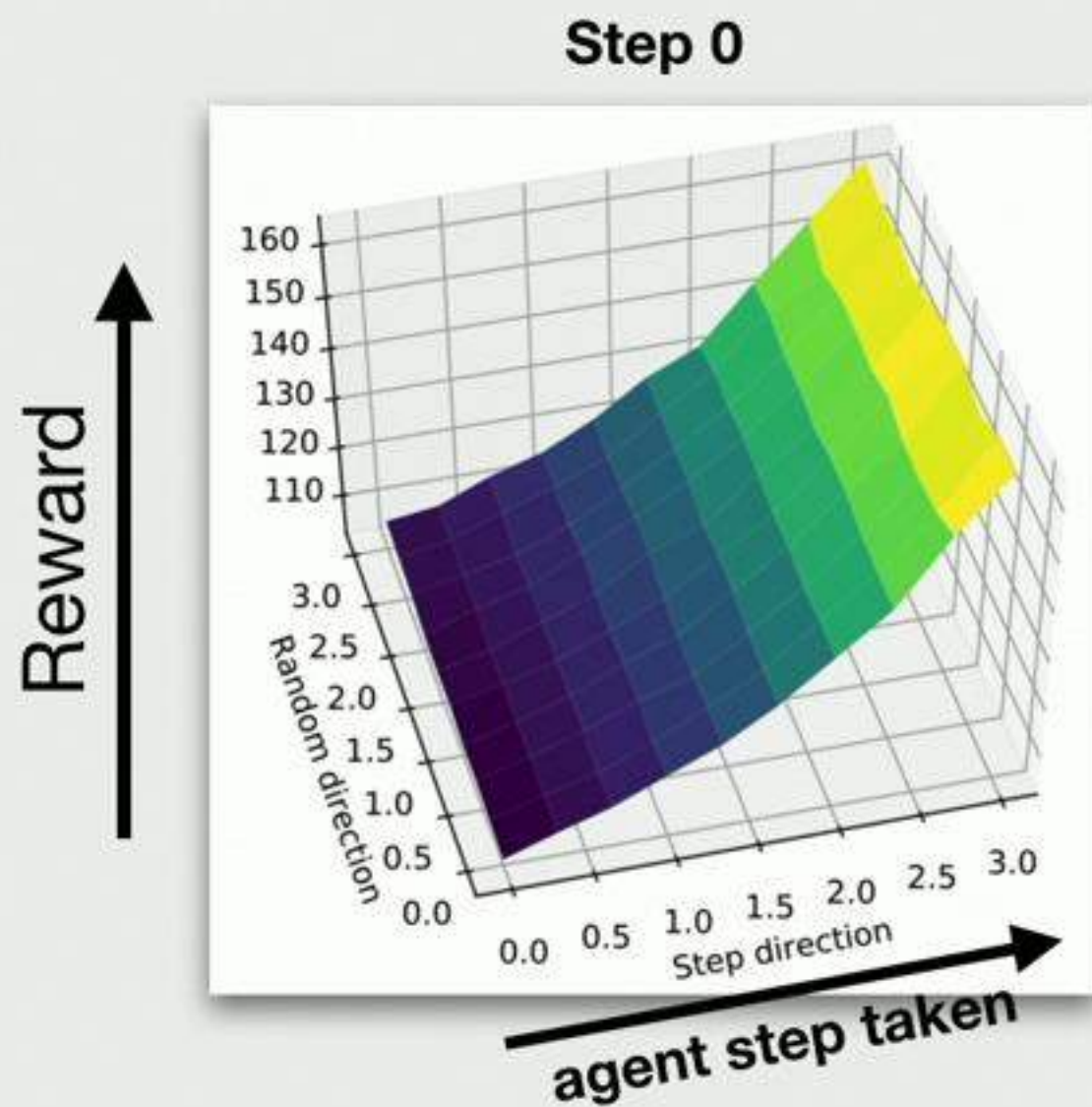
Optimization Landscapes



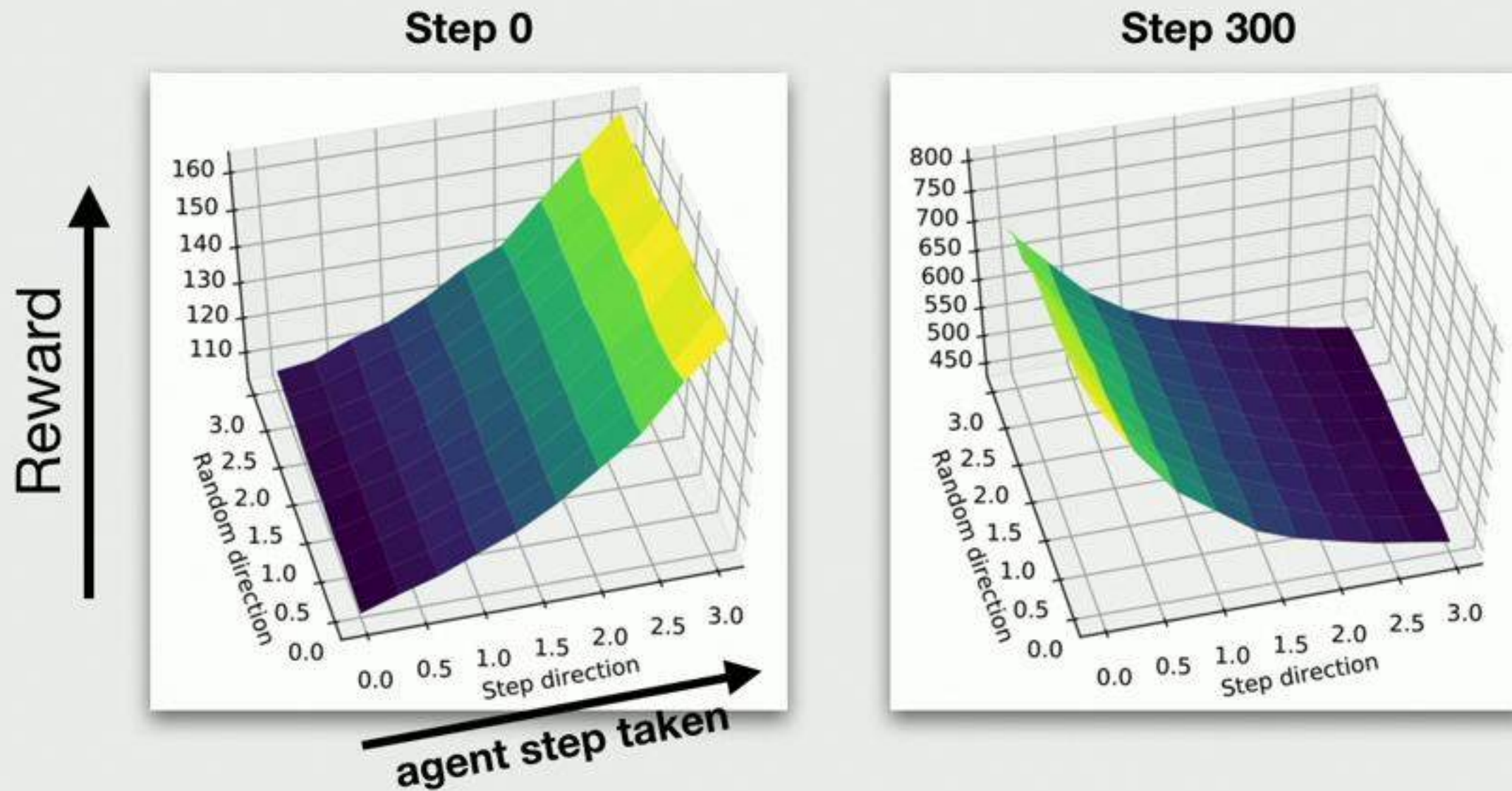
Optimization Landscapes



Optimization Landscapes

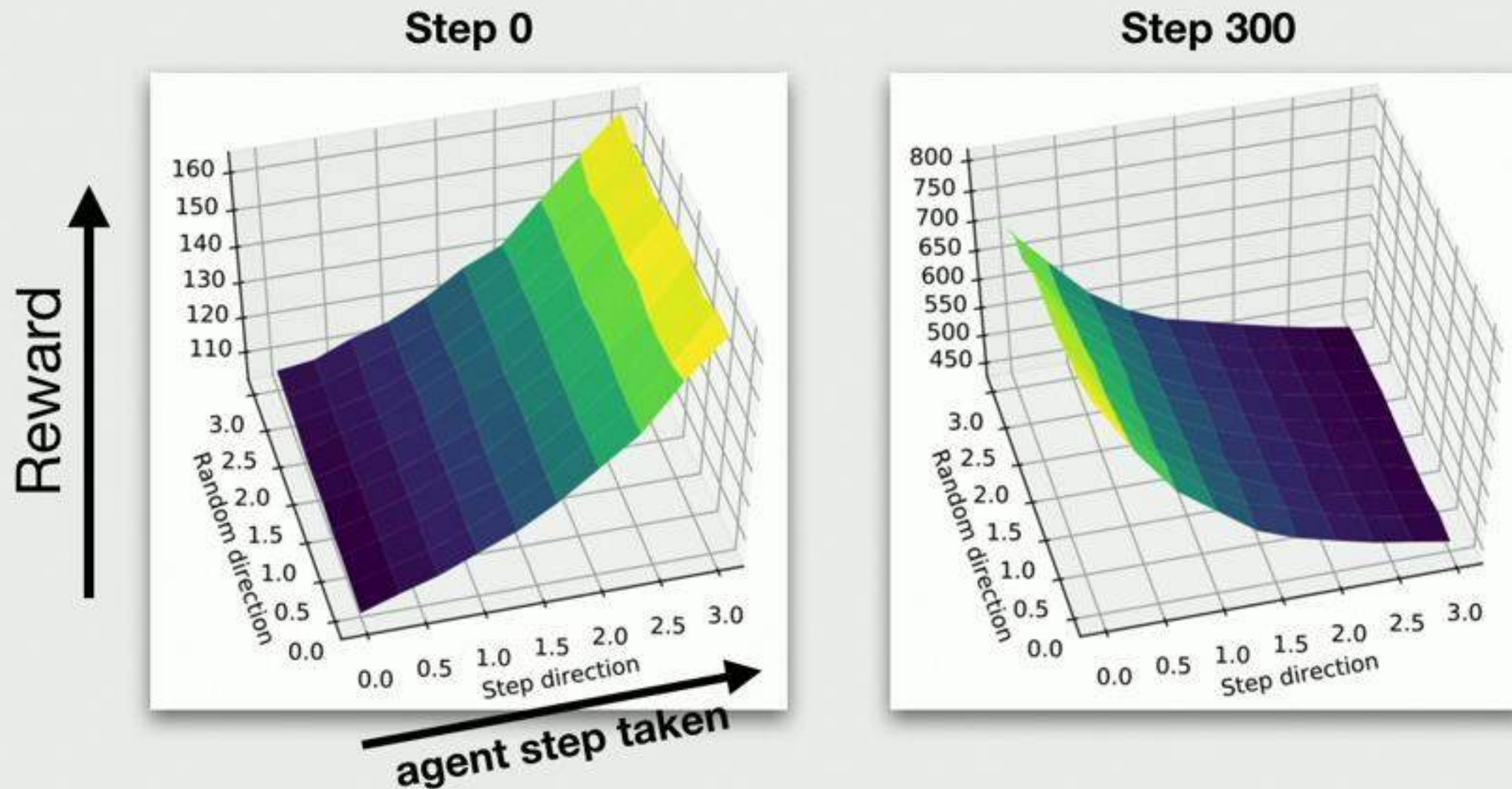


Optimization Landscapes



Steps are often not predictive

Optimization Landscapes



Steps are often not predictive

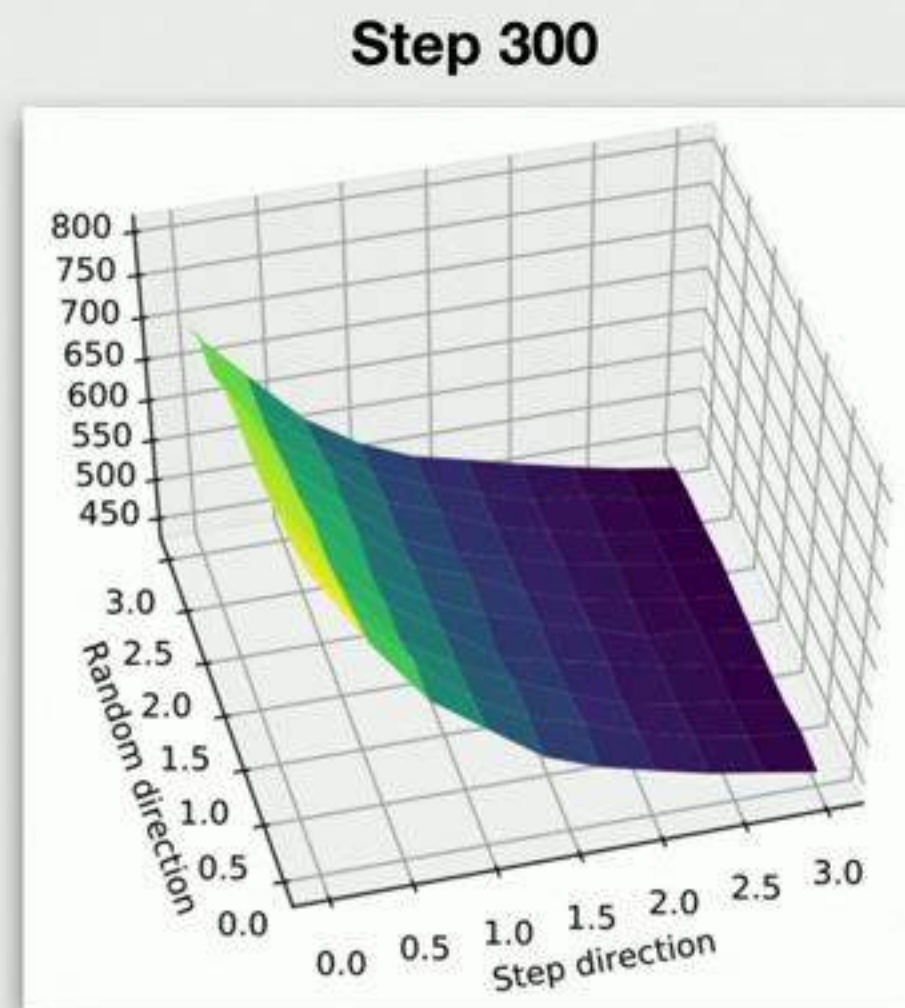
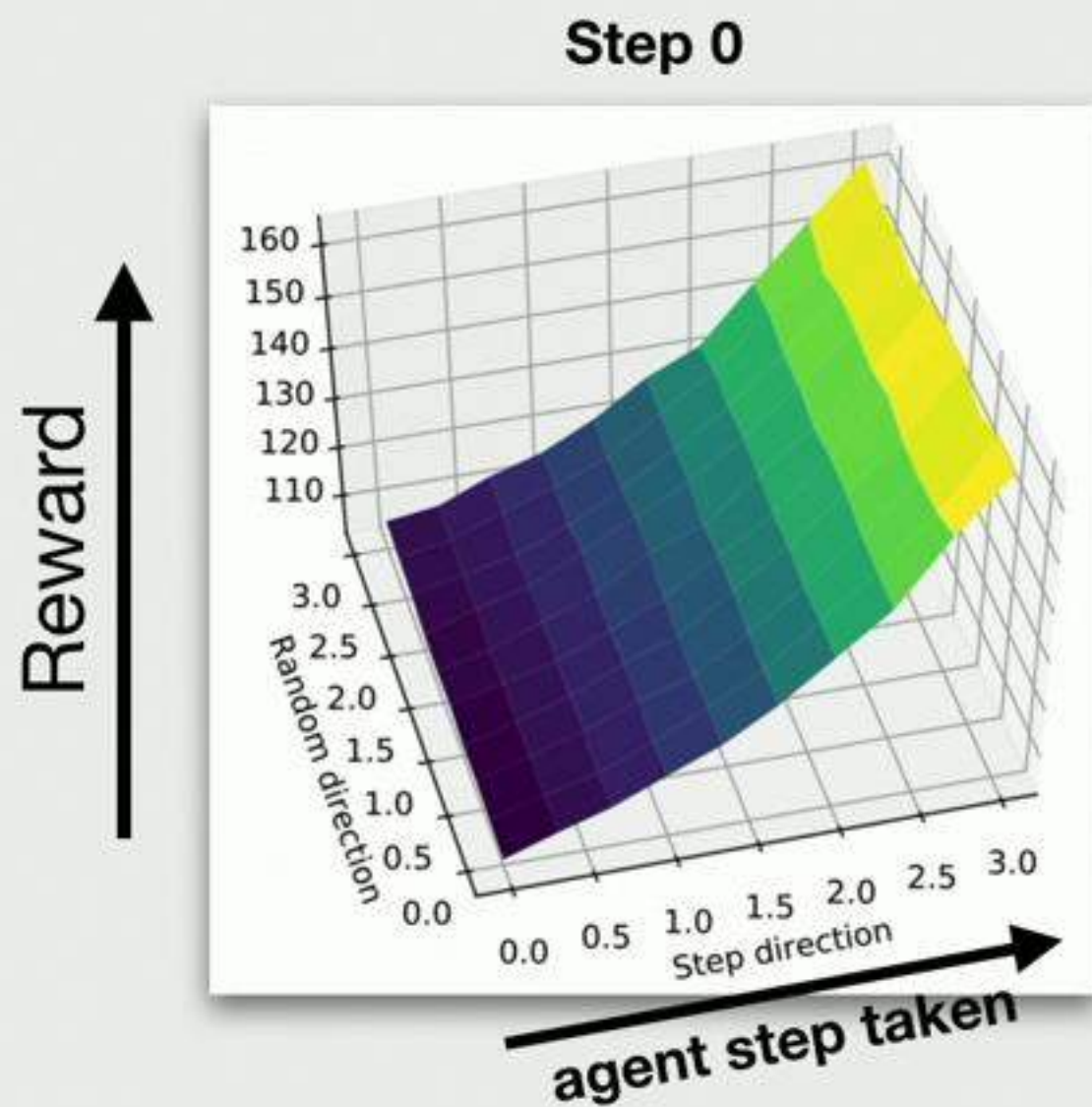
What's going on here?

Optimization Landscapes

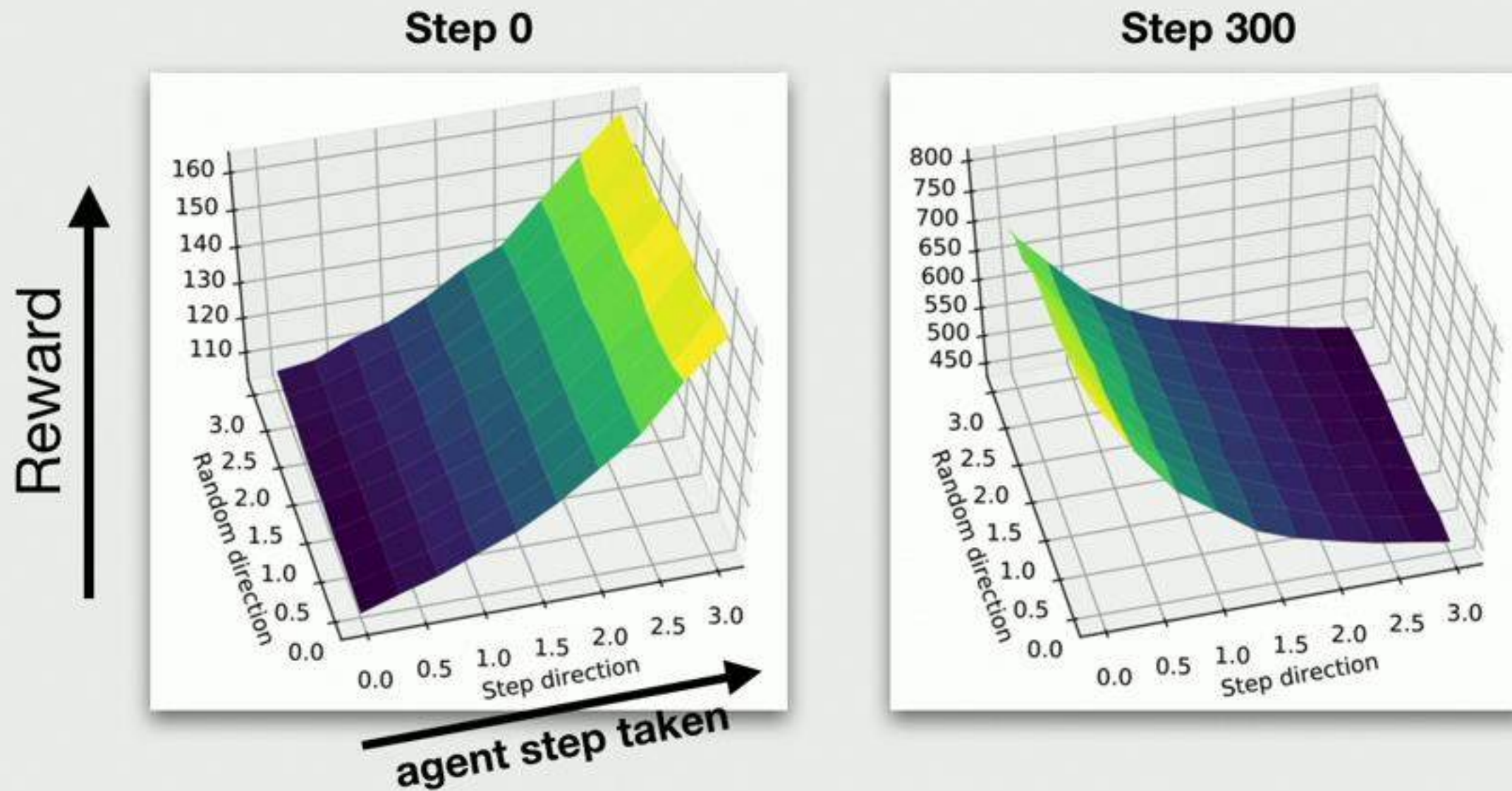
Methods iteratively maximize a “surrogate reward”

(not the true reward!)

Optimization Landscapes

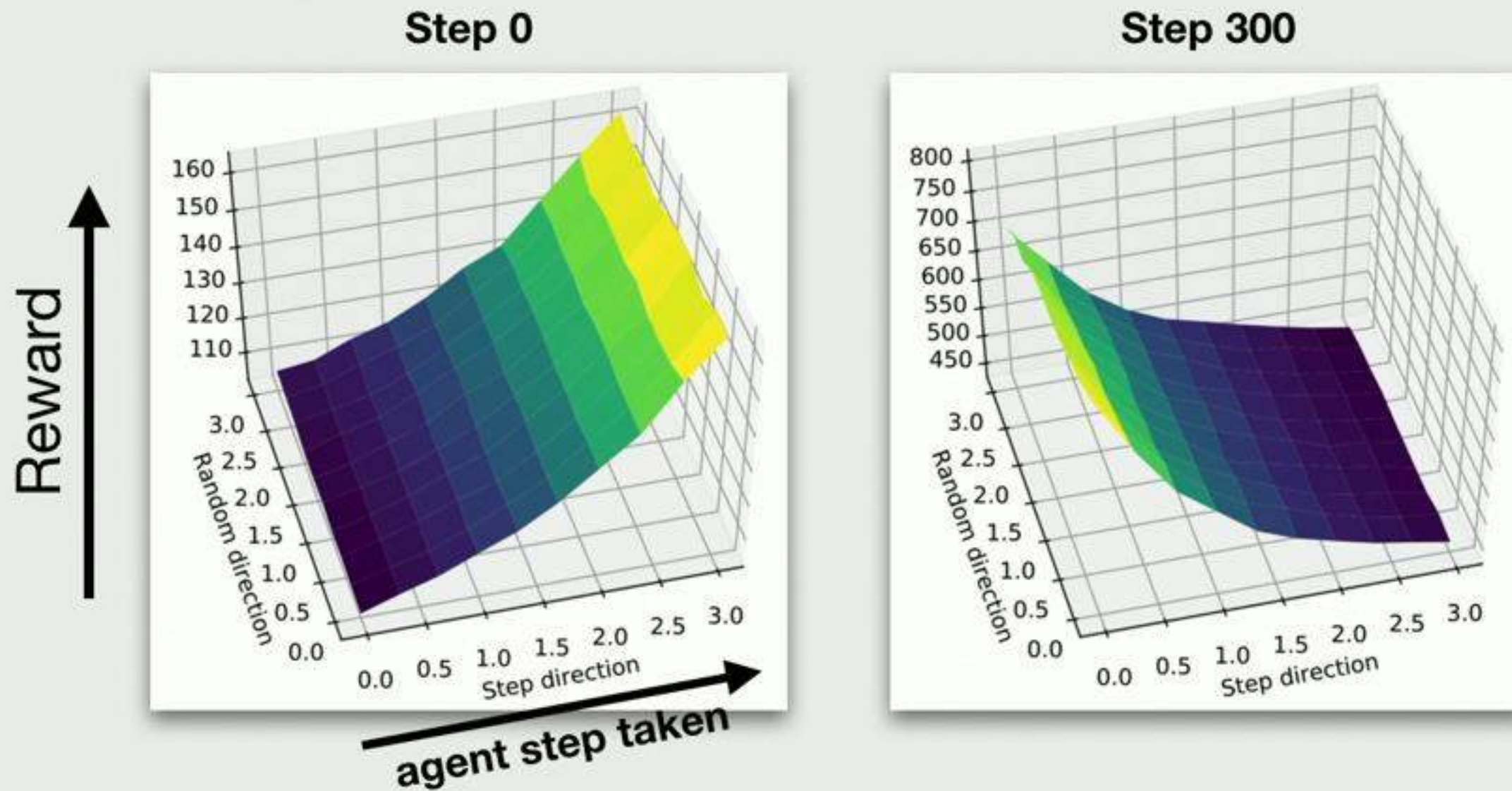


Optimization Landscapes



Steps are often not predictive

Optimization Landscapes



Steps are often not predictive

What's going on here?

Optimization Landscapes

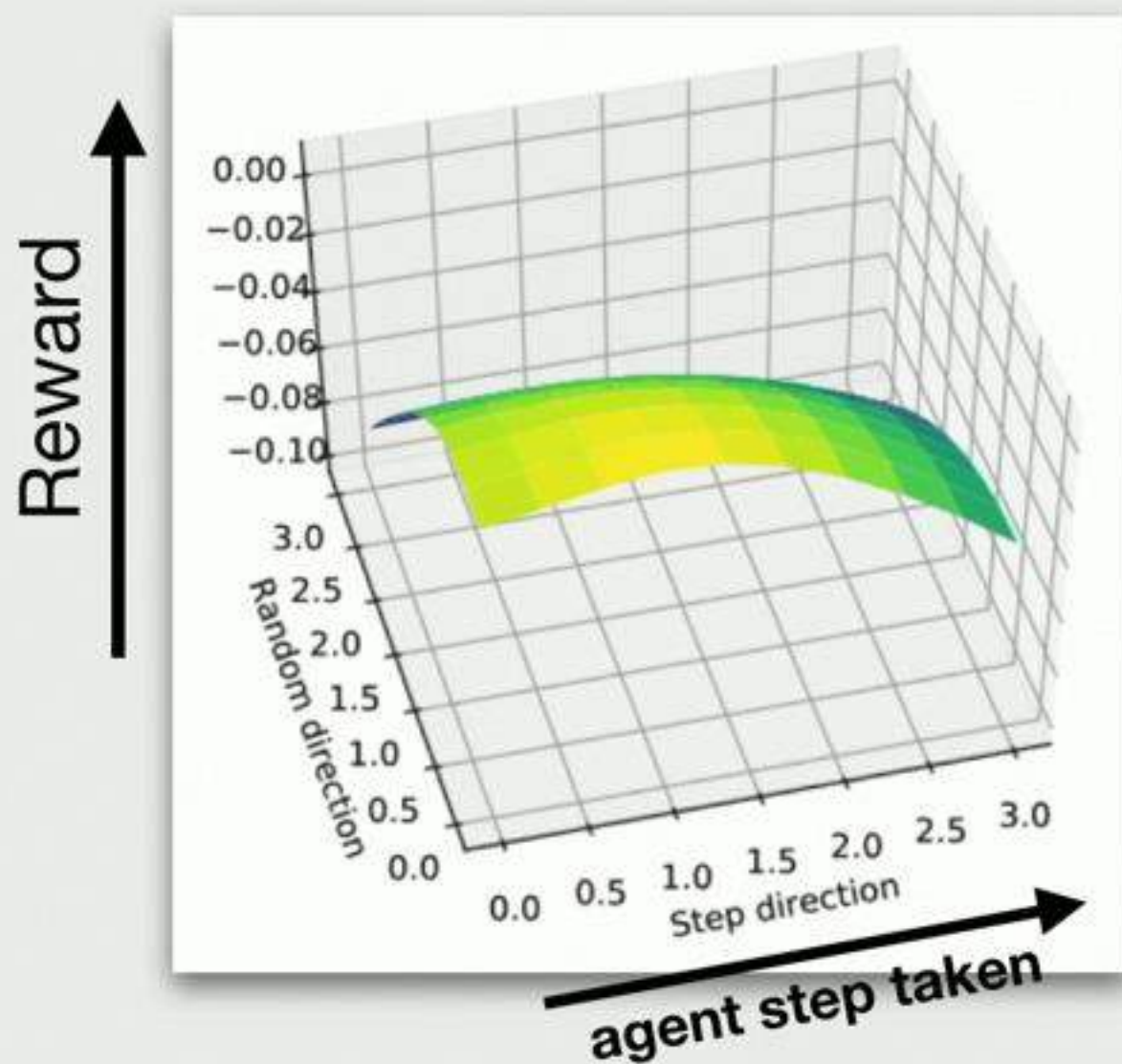
Methods iteratively maximize a “surrogate reward”

(not the true reward!)

How do **surrogate rewards** compare with **true rewards**?

Optimization Landscapes

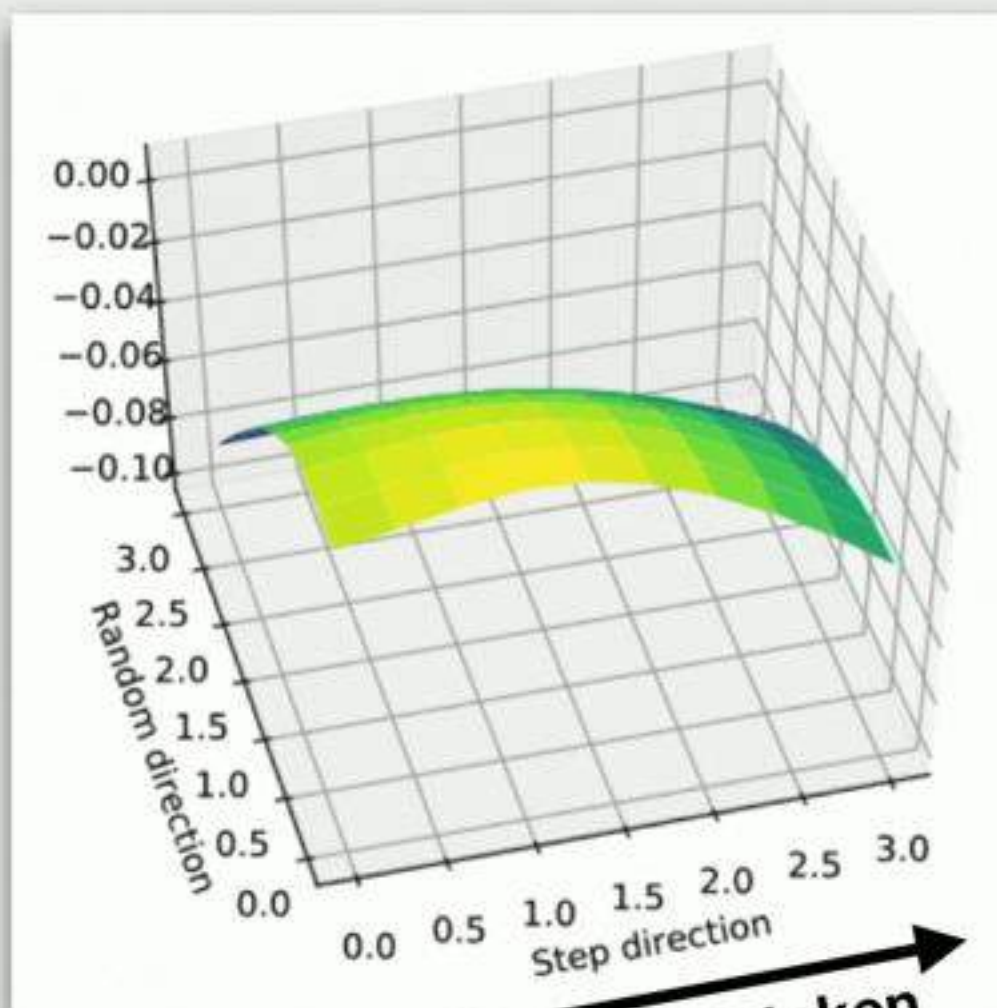
Surrogate Landscape



Optimization Landscapes

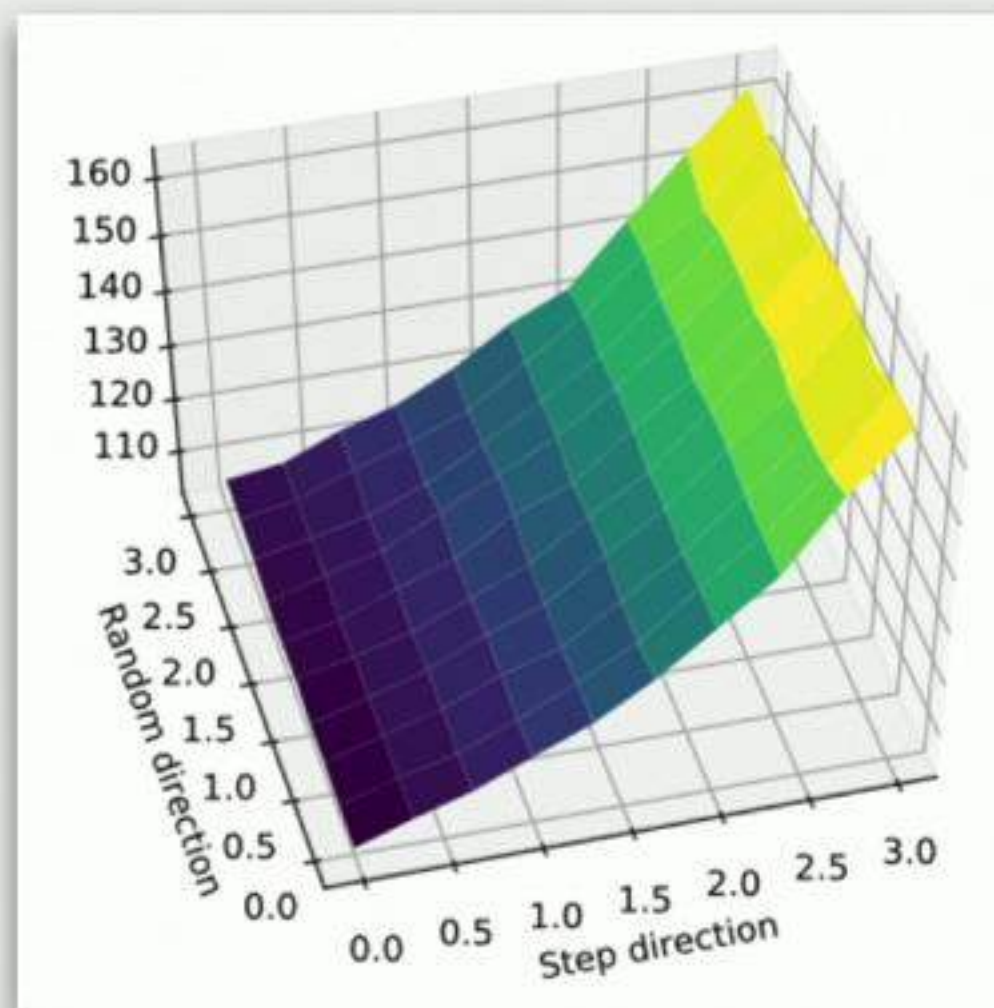
Surrogate Landscape

Reward ↑



agent step taken →

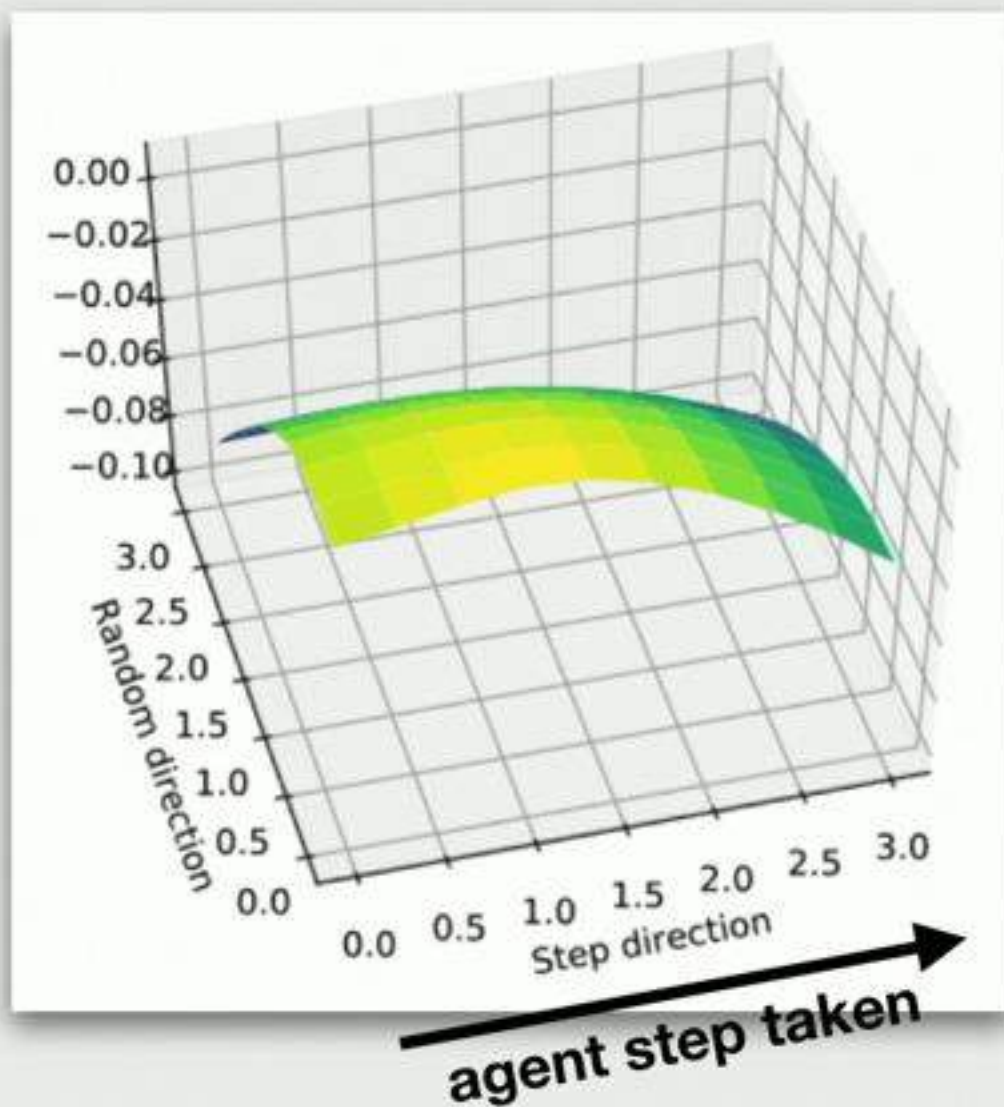
Reward Landscape



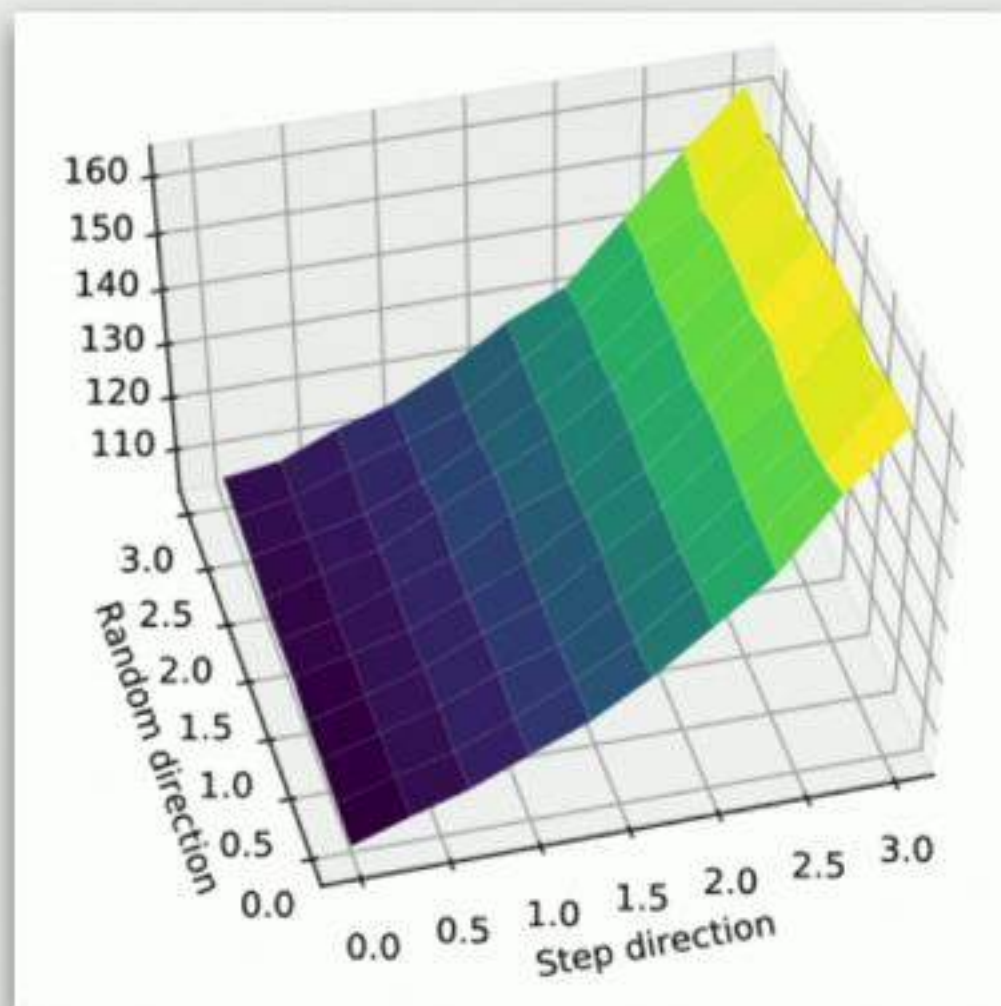
Optimization Landscapes

Surrogate Landscape

Reward ↑



Reward Landscape

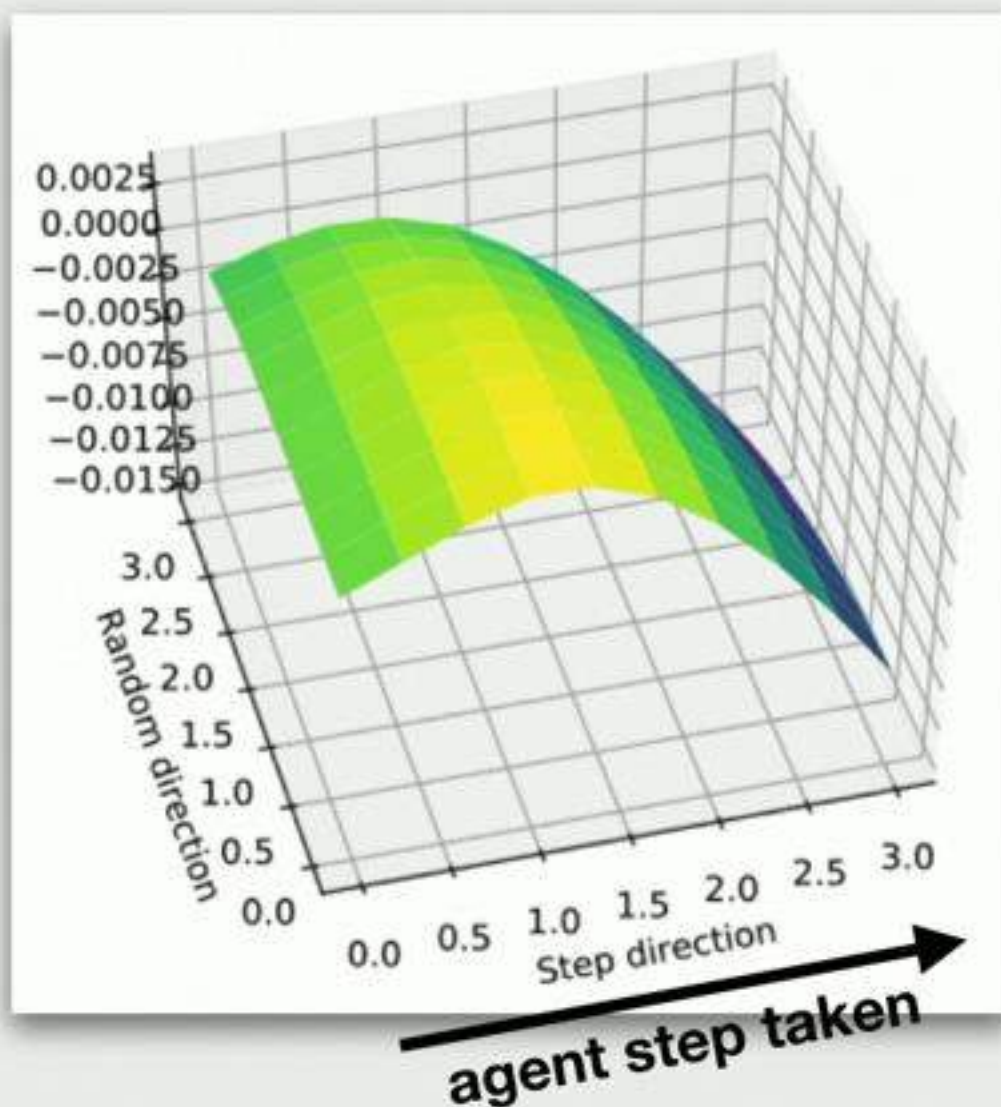


Step 0

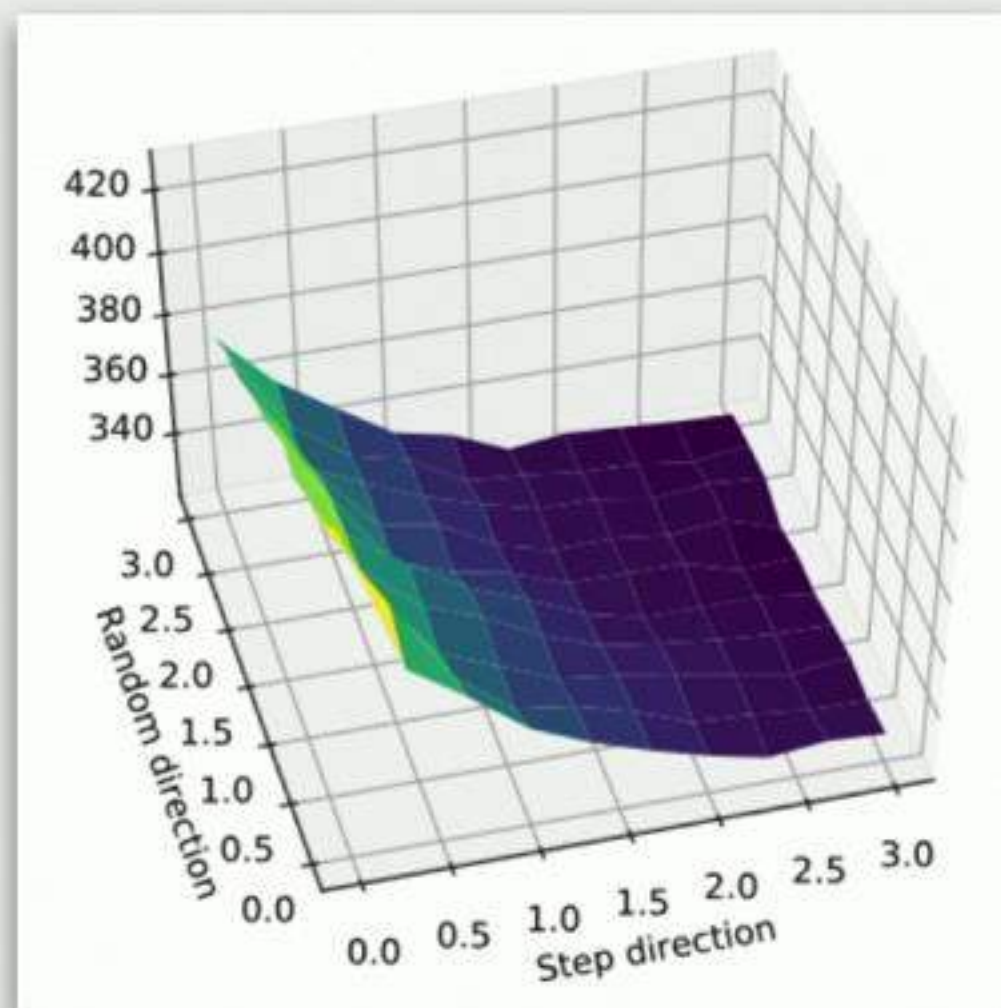
Optimization Landscapes

Surrogate Landscape

Reward ↑



Reward Landscape

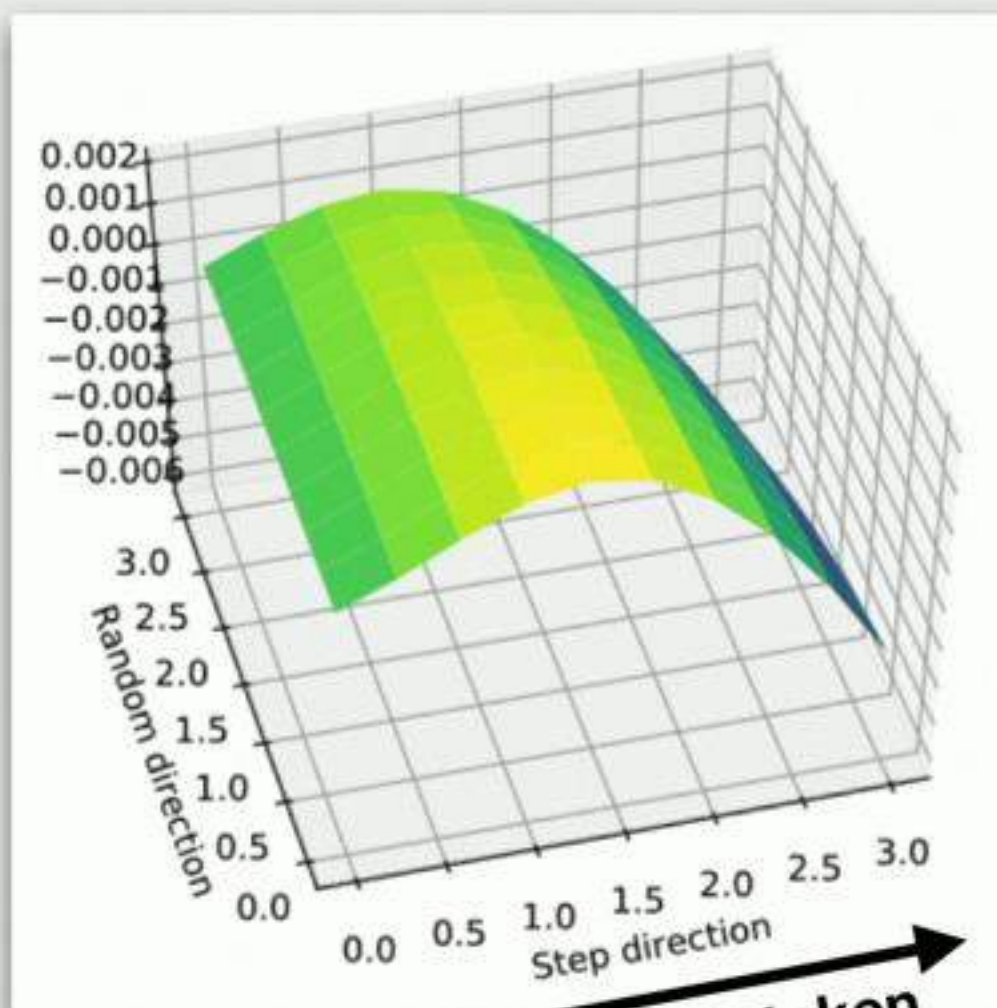


Step 150

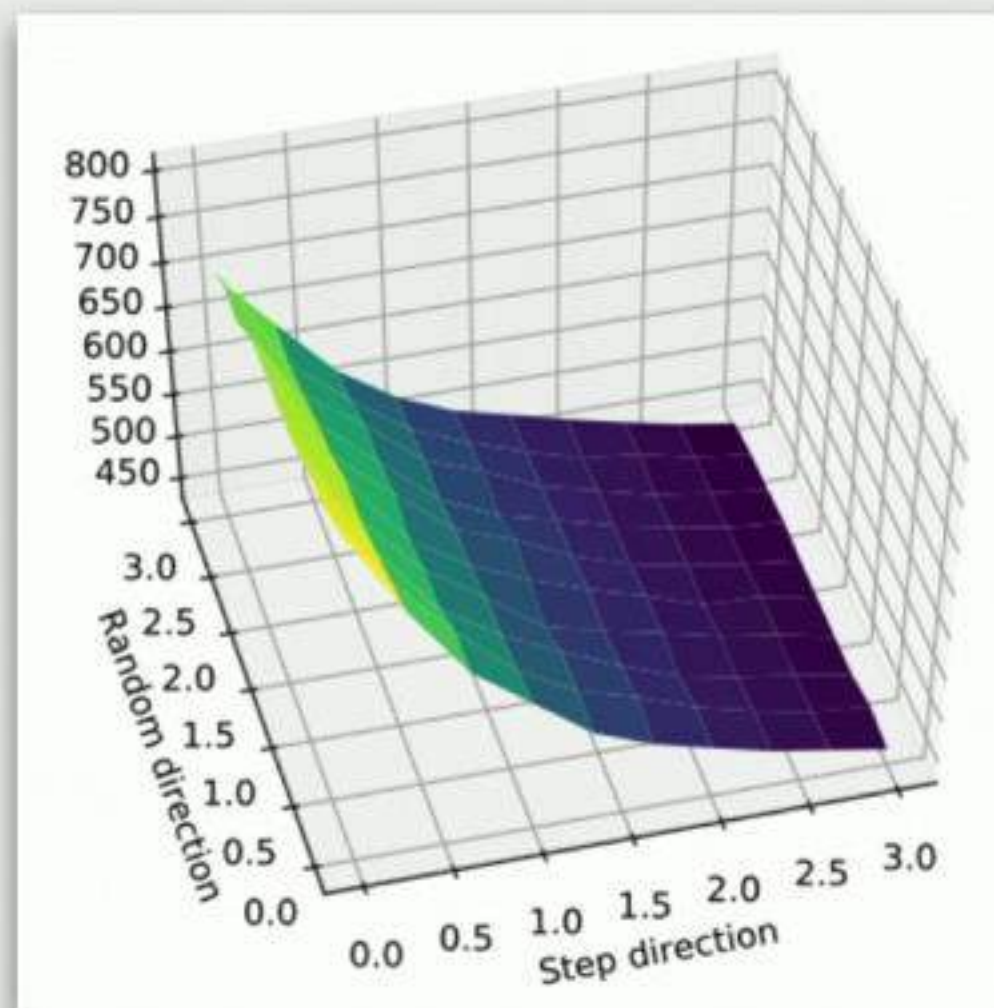
Optimization Landscapes

Surrogate Landscape

Reward ↑



Reward Landscape

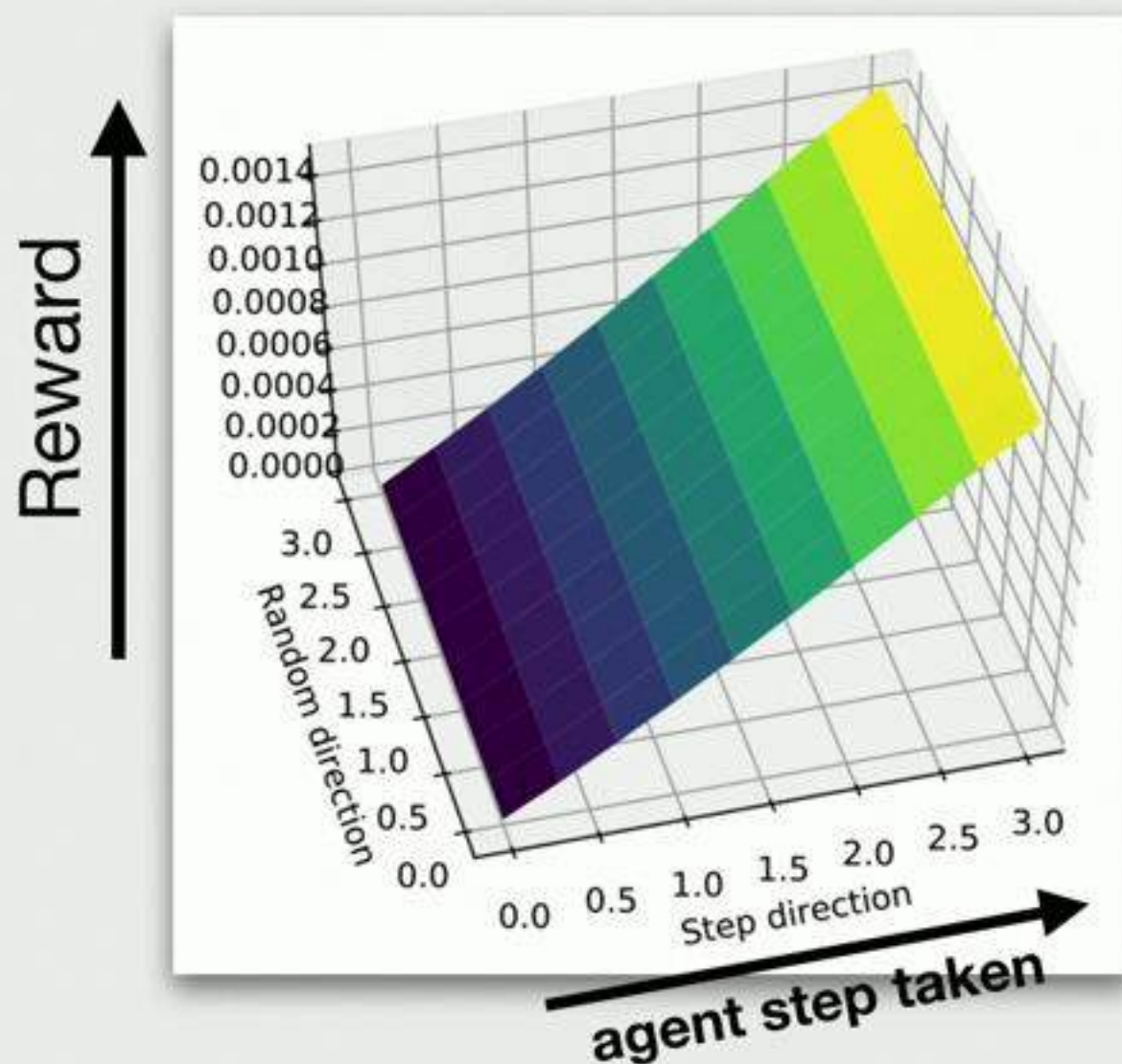


agent step taken →

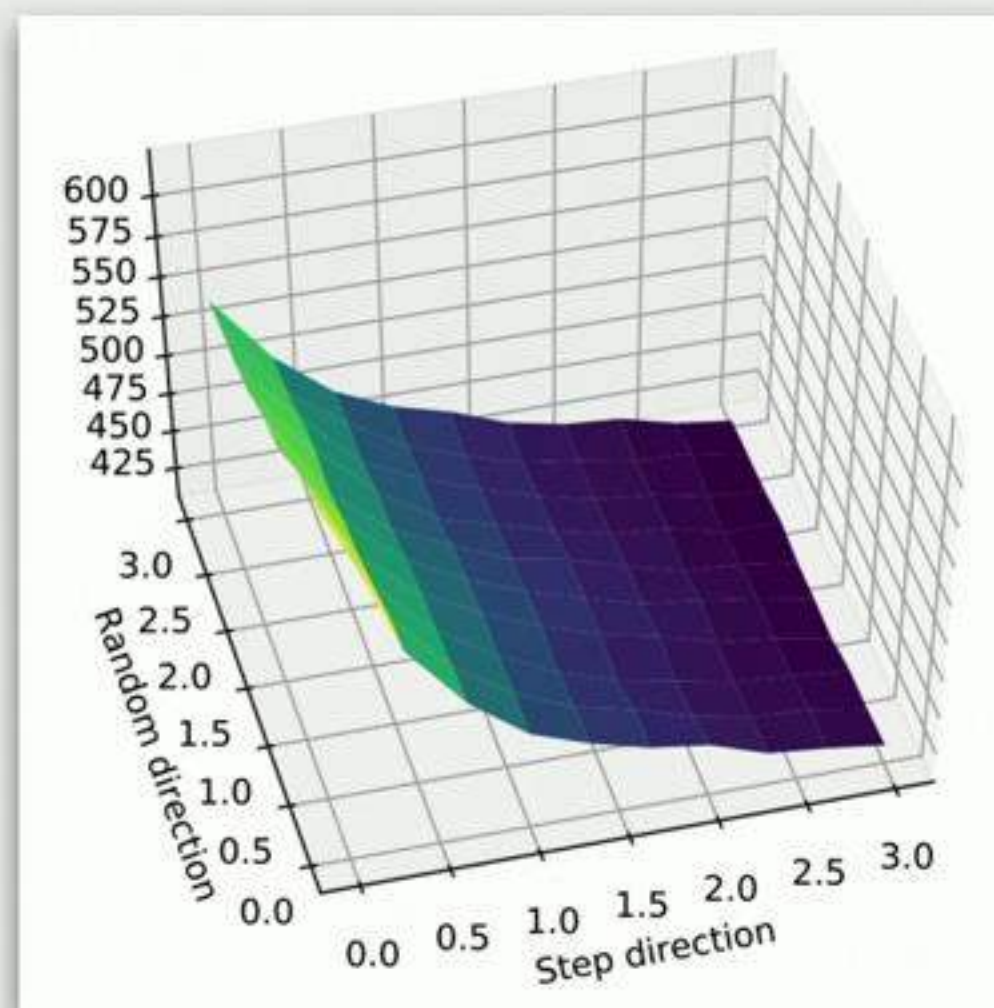
Step 300

Optimization Landscapes

Surrogate Landscape



Reward Landscape

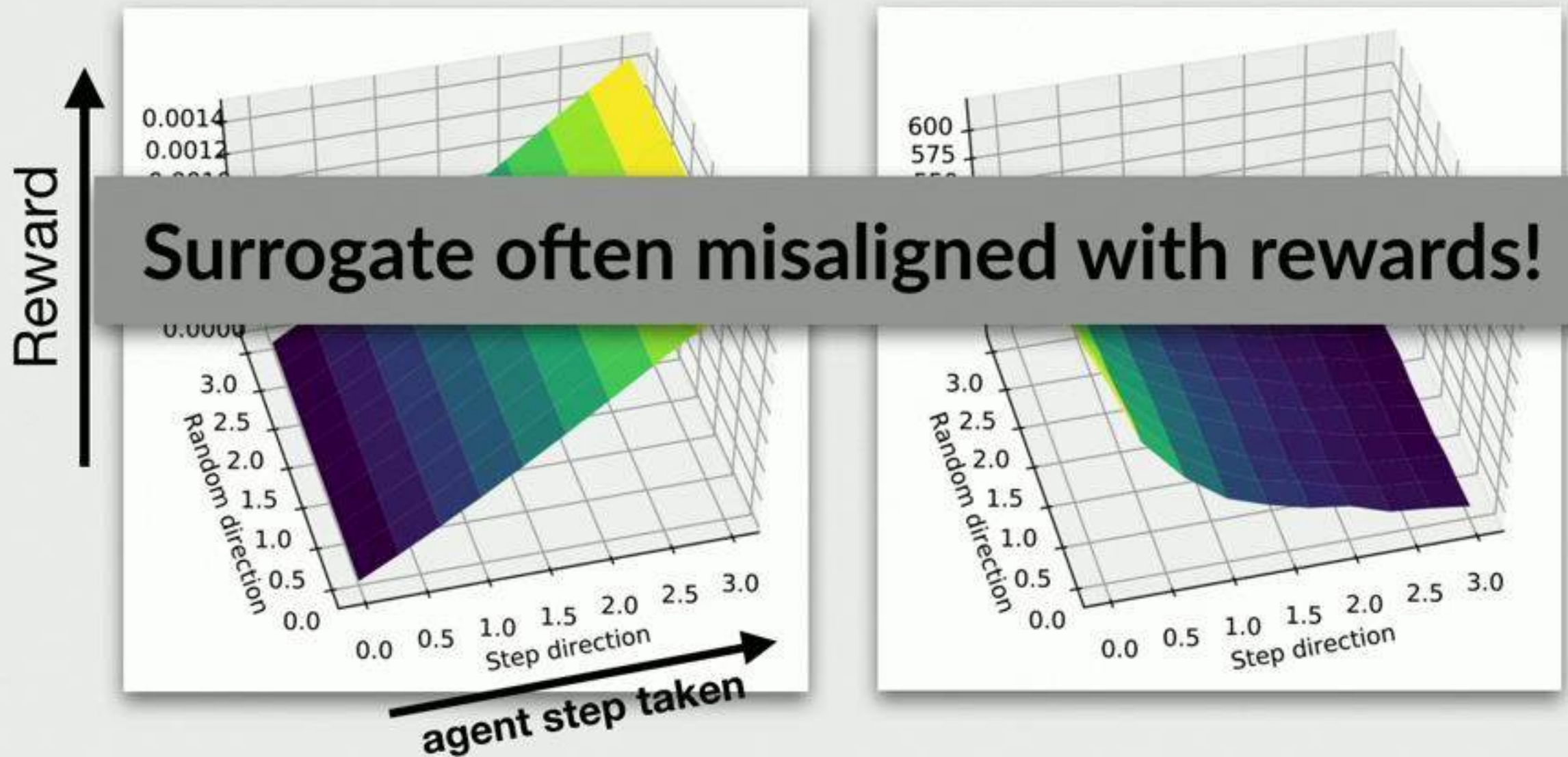


Step 450

Optimization Landscapes

Surrogate Landscape

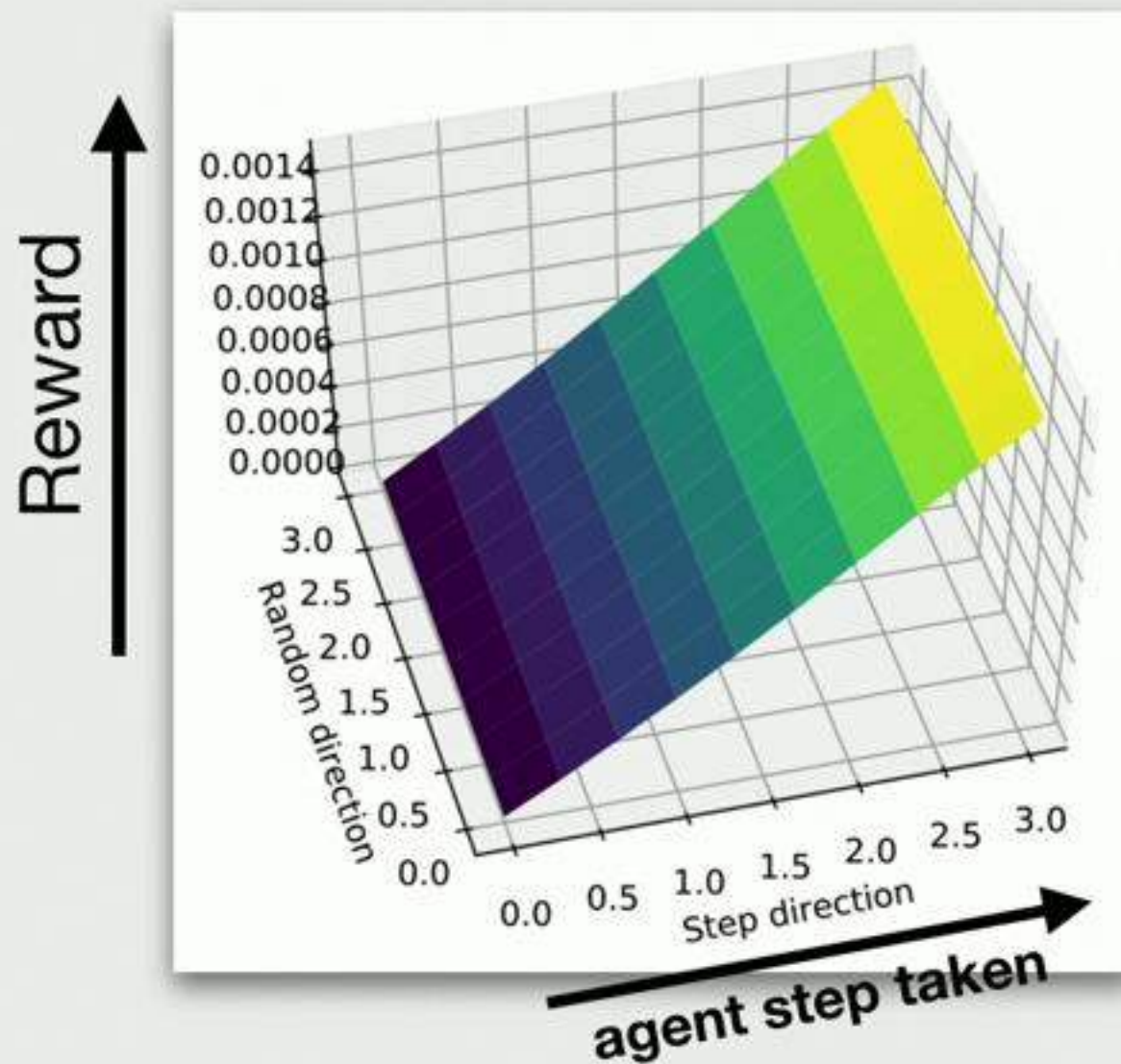
Reward Landscape



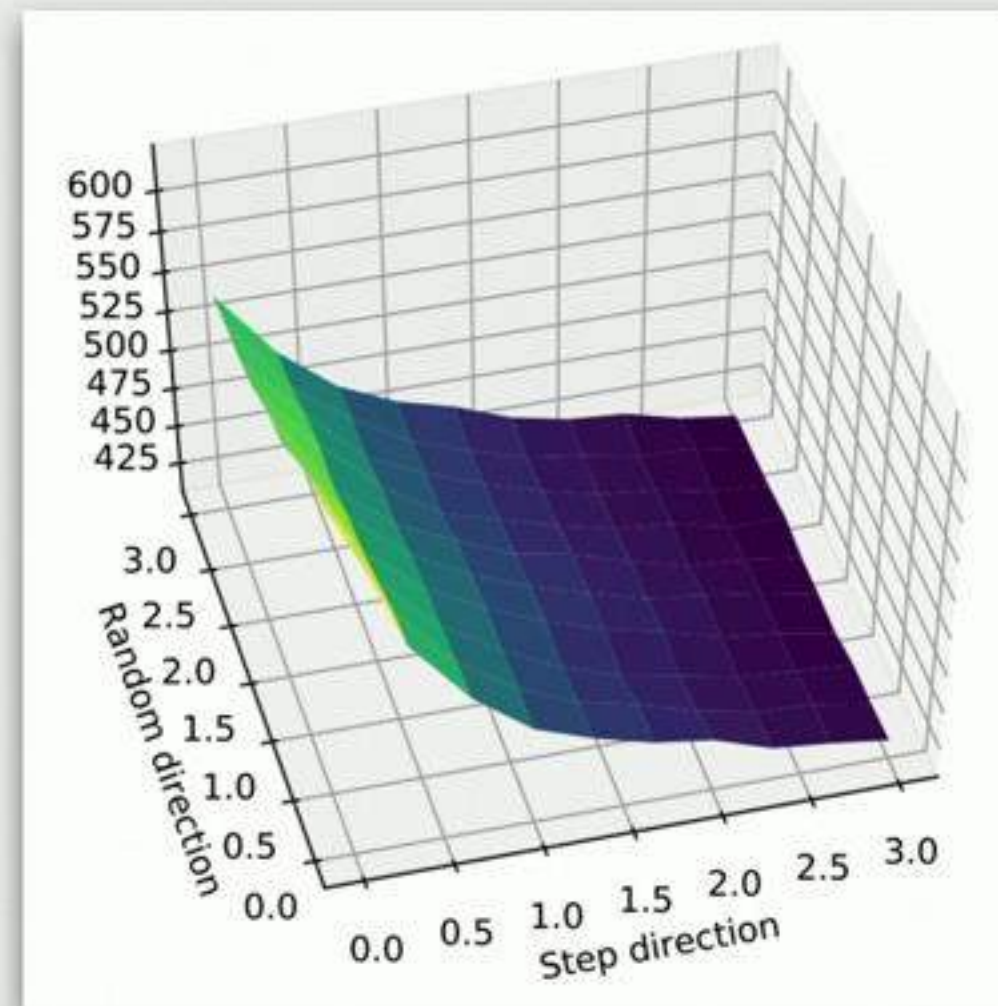
Step 450

Optimization Landscapes

Surrogate Landscape



Reward Landscape

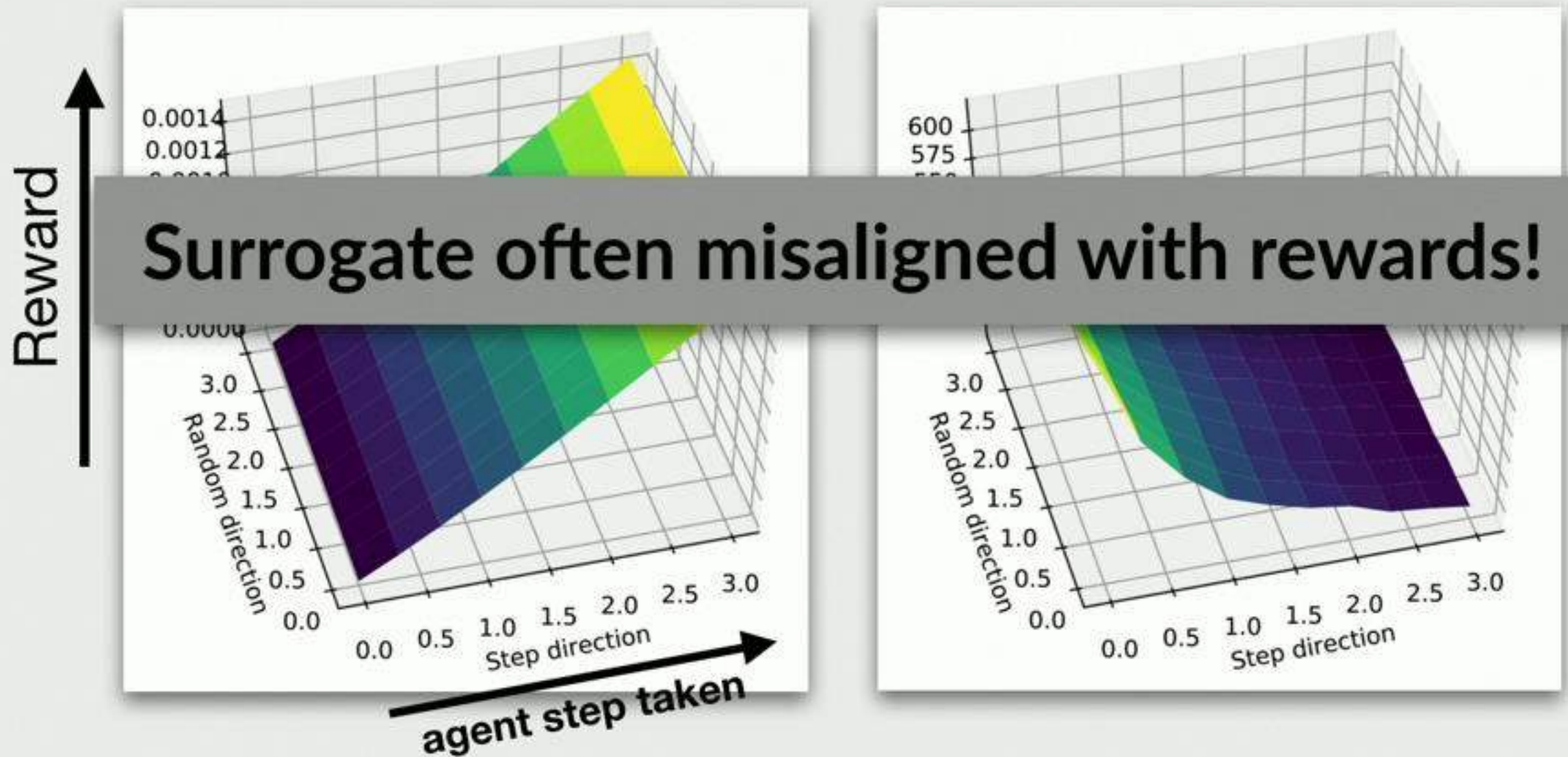


Step 450

Optimization Landscapes

Surrogate Landscape

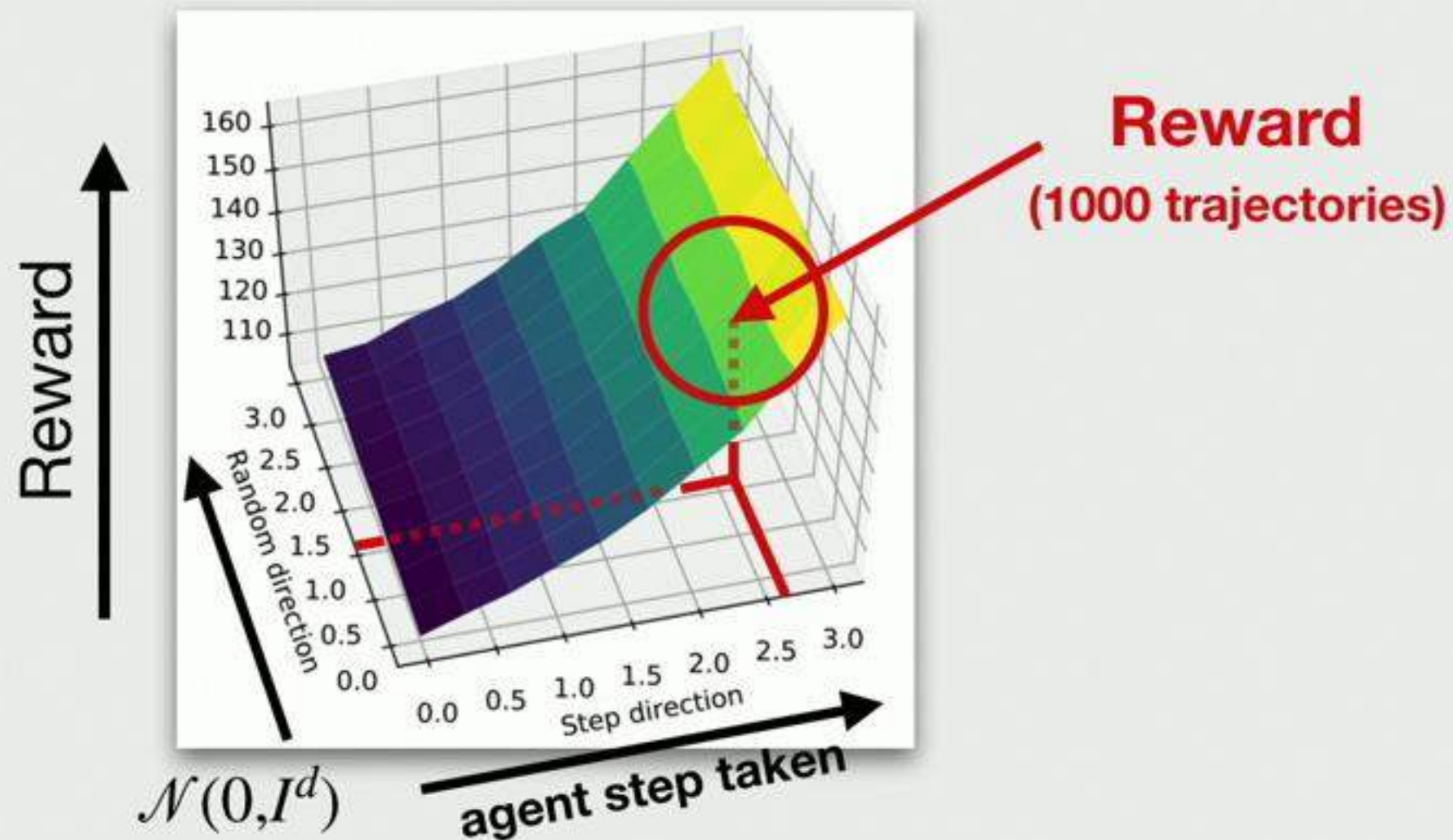
Reward Landscape



Step 450

Optimization Landscapes

All landscapes so far are in the **high sample regime**

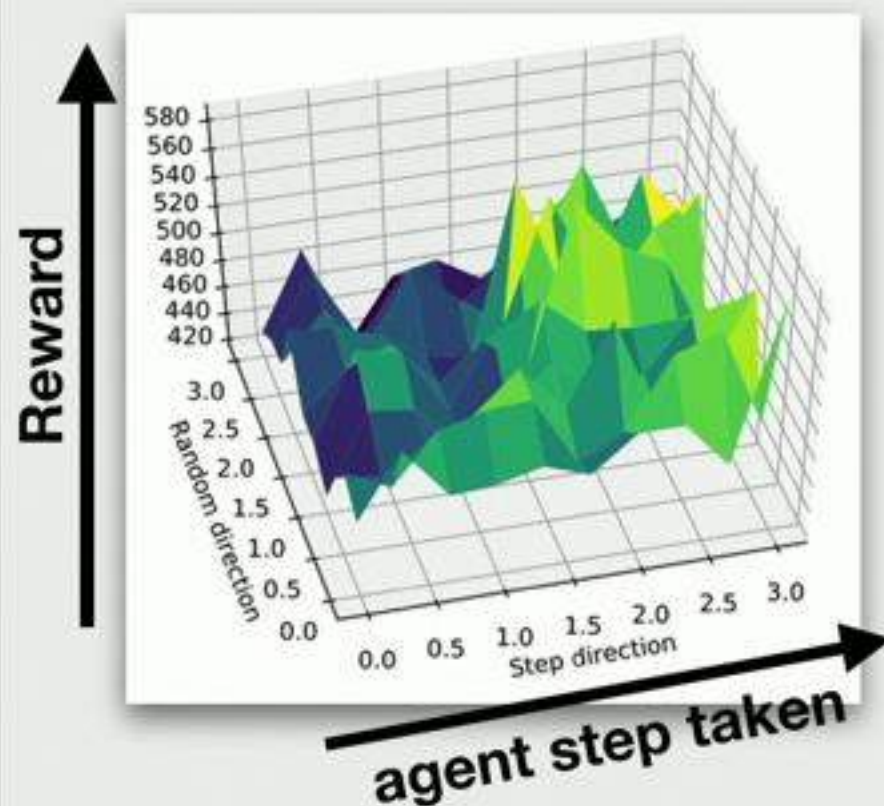


How do landscapes appear to the agent?

(~20 trajectories)

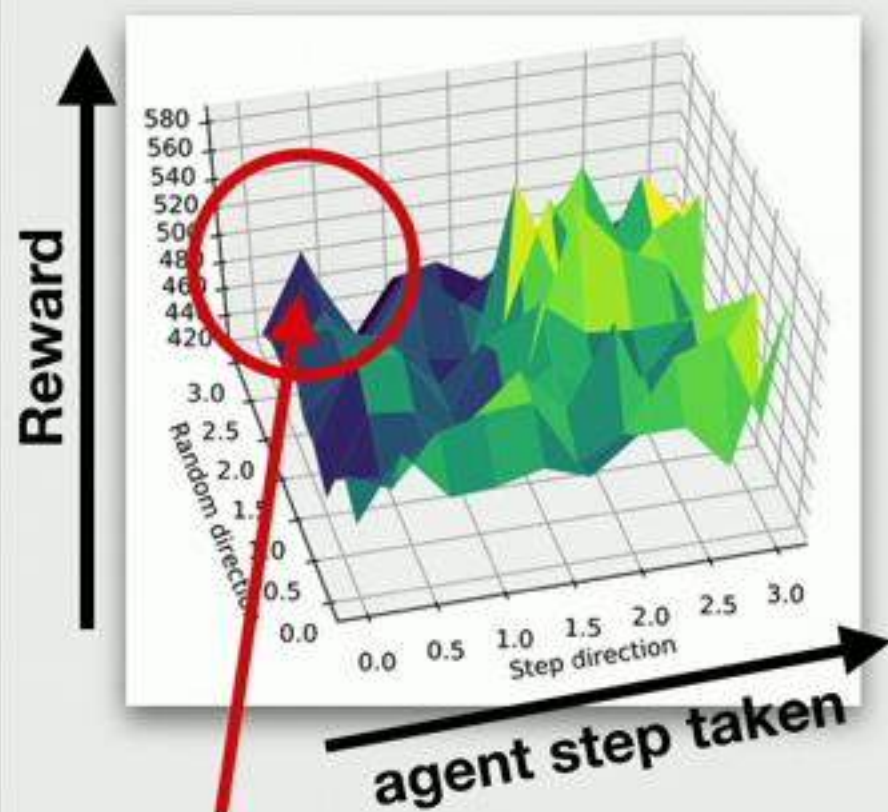
Optimization Landscapes

20-sample estimates



Optimization Landscapes

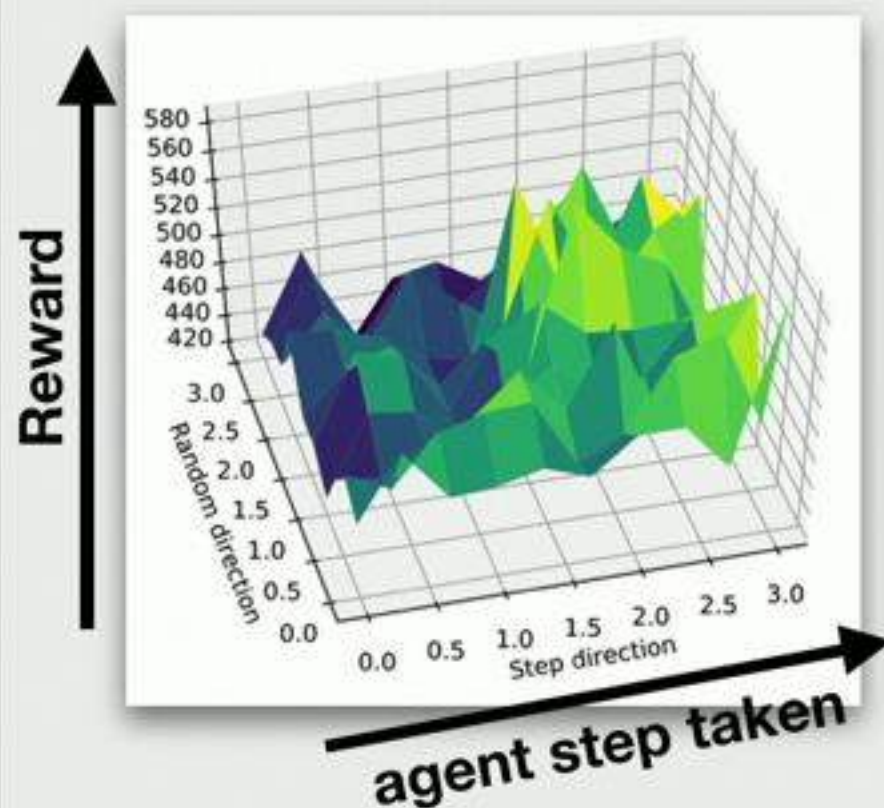
20-sample estimates



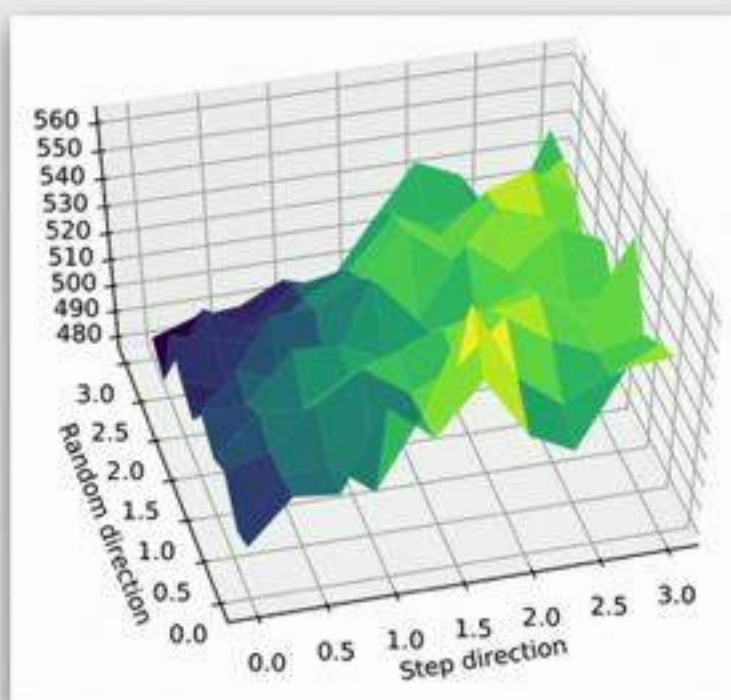
20 trajectories per reward estimate

Optimization Landscapes

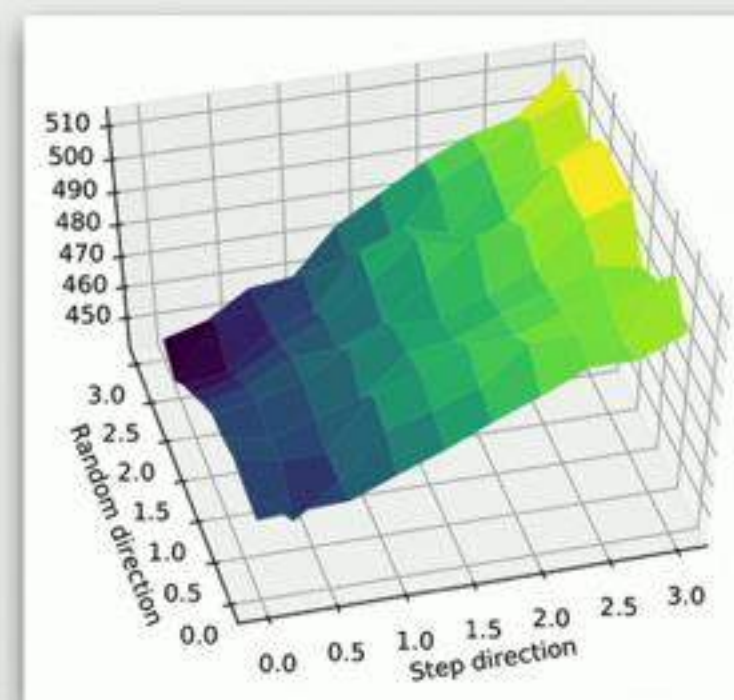
20-sample estimates



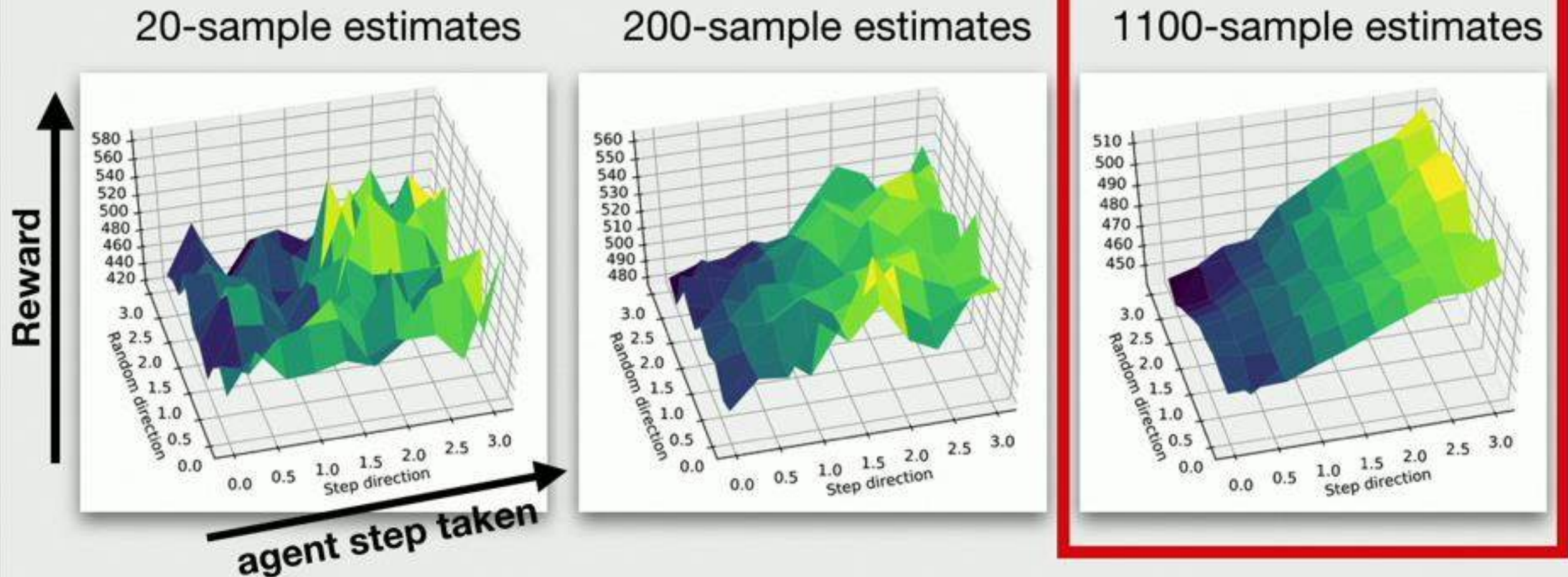
200-sample estimates



1000-sample estimates

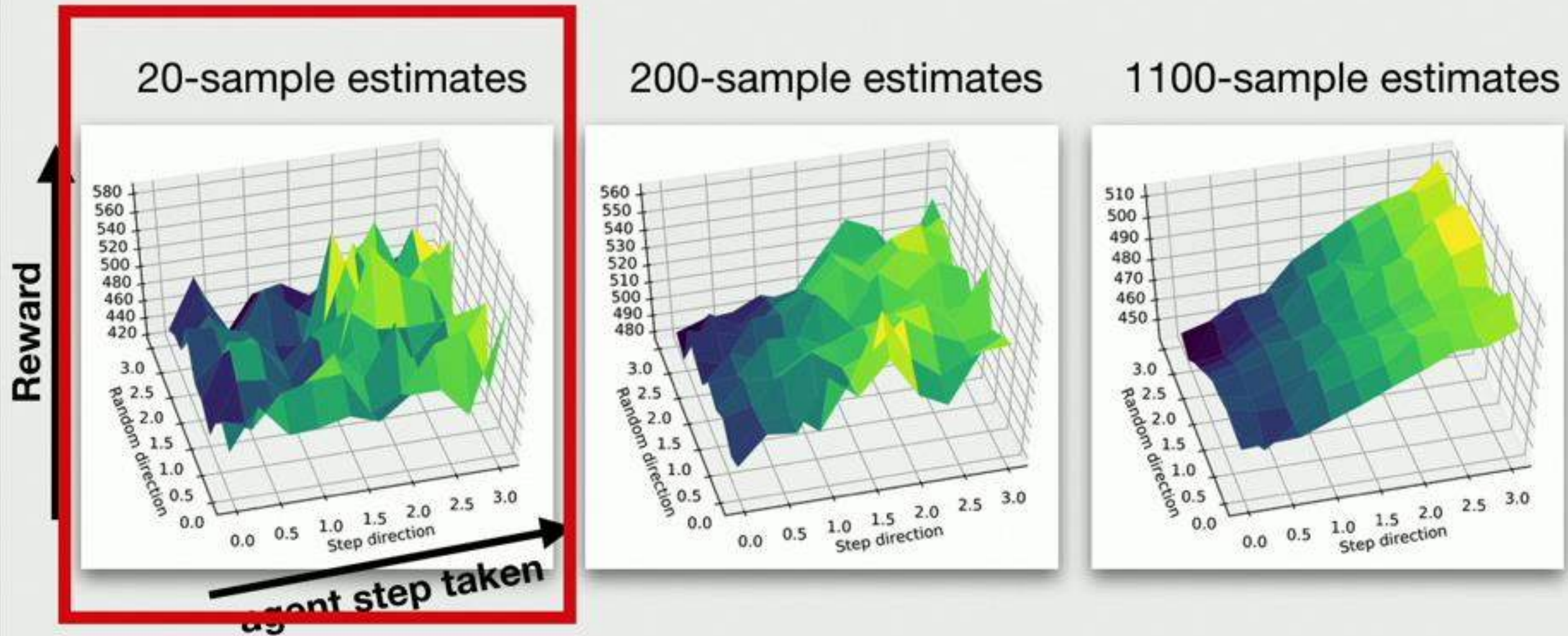


Optimization Landscapes



using many samples induces a smooth landscape...

Optimization Landscapes



using many samples induces a smooth landscape...

... but improvement is hard to detect in the agent's sample regime

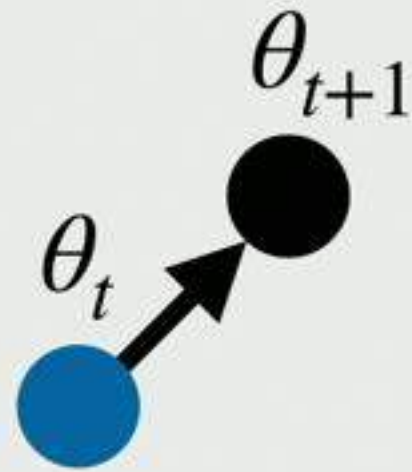
Optimization Landscapes

- ▶ Surrogate landscapes are often not reflective of rewards
- ▶ How can we better navigate the reward landscape?

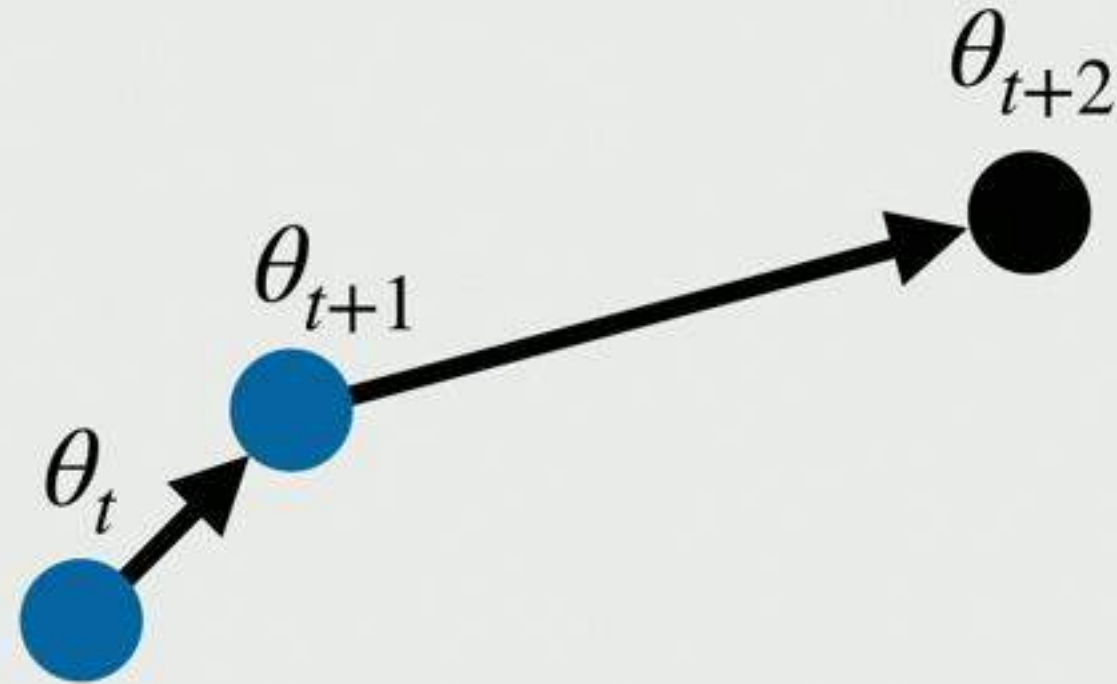
Trust Regions



Trust Regions



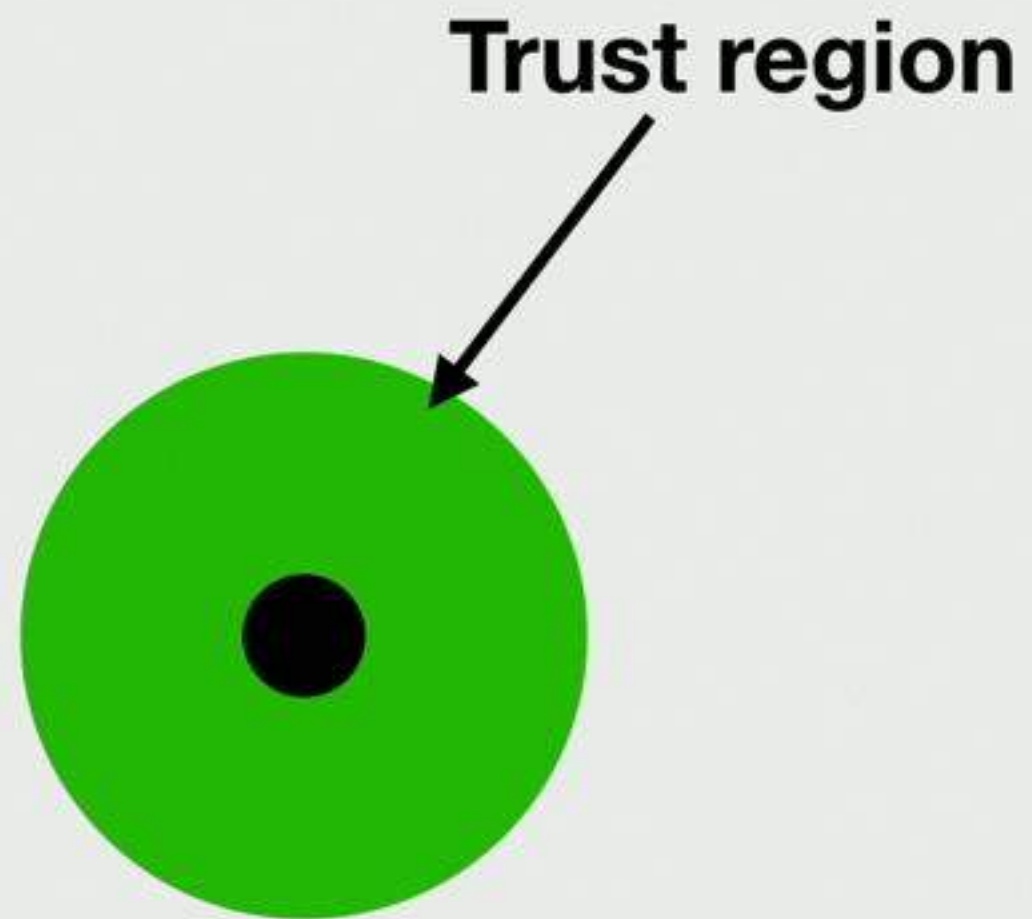
Trust Regions



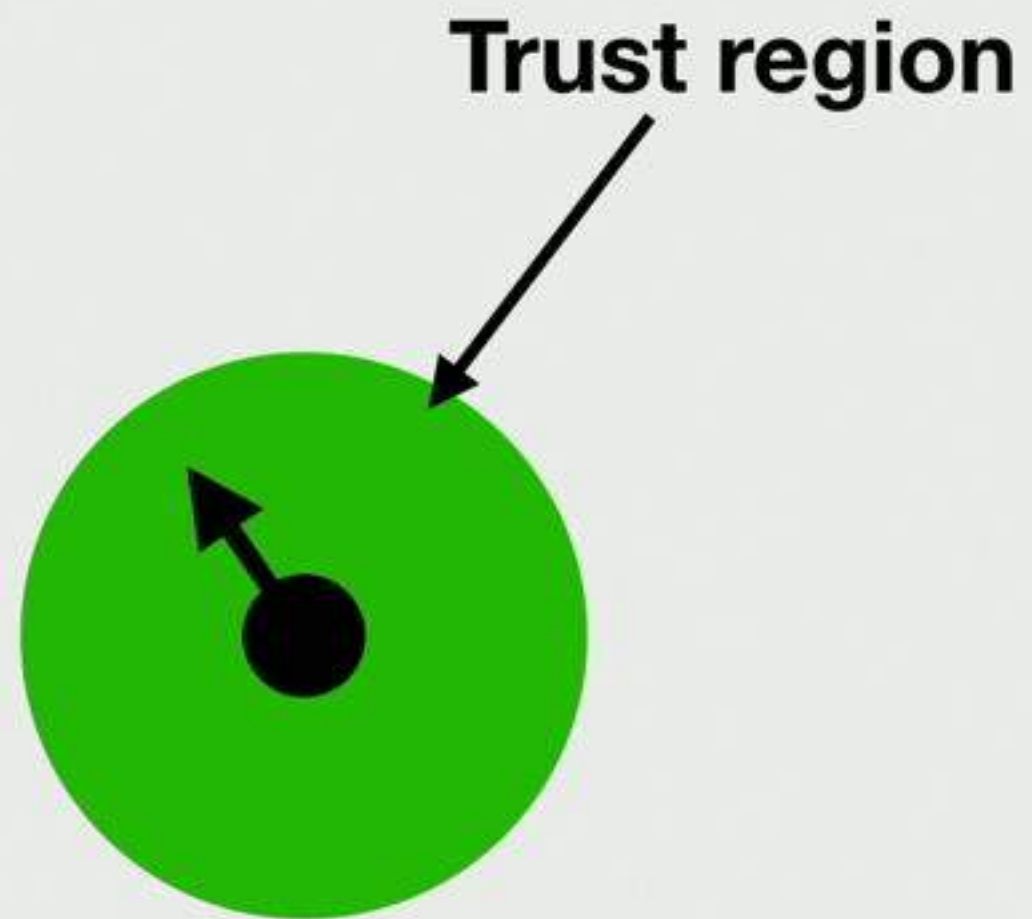
Trust Regions



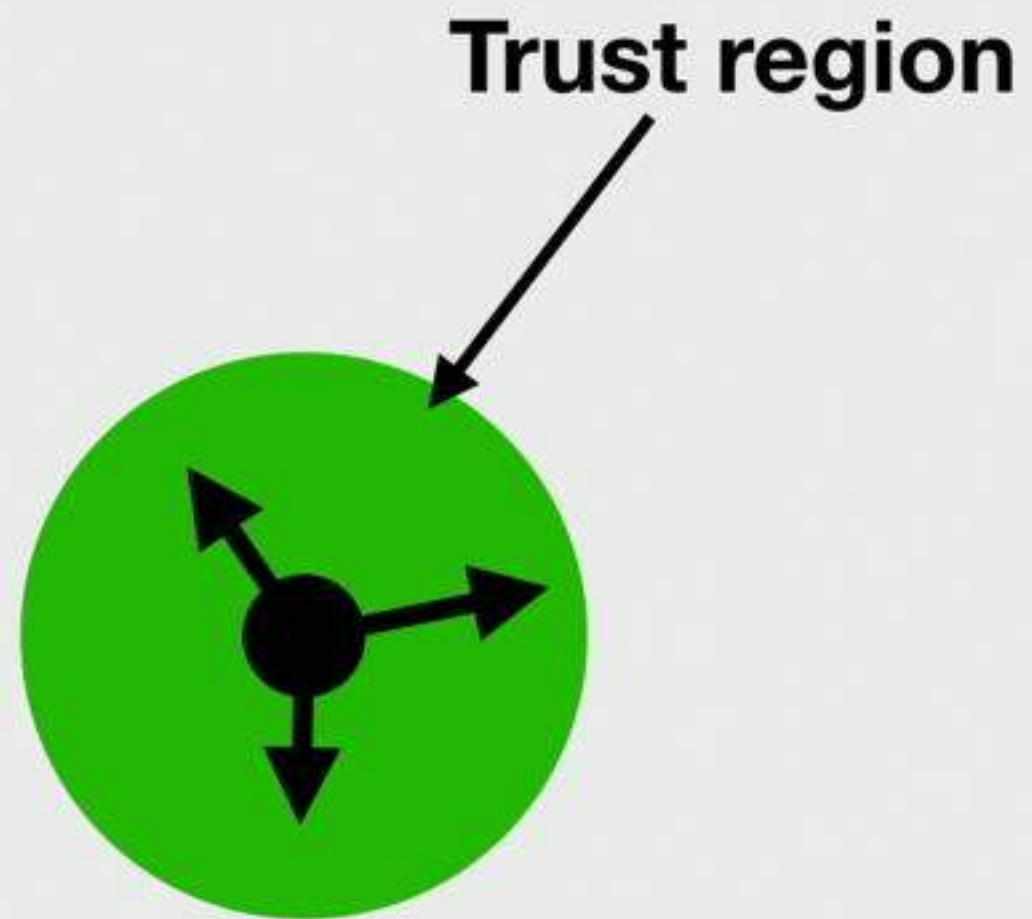
Trust Regions



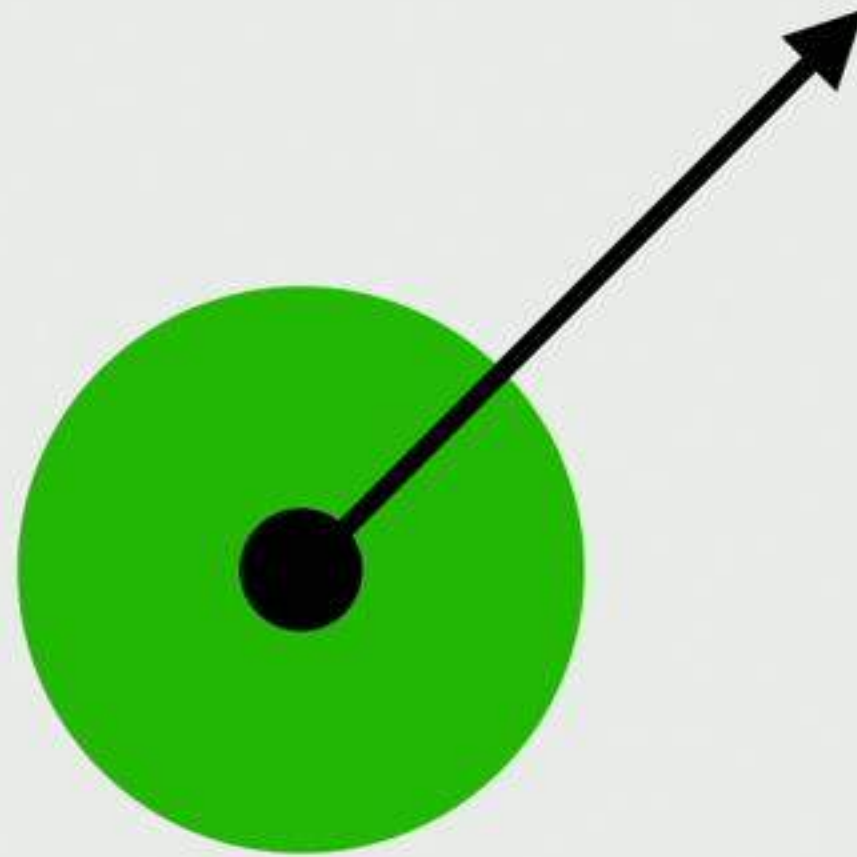
Trust Regions



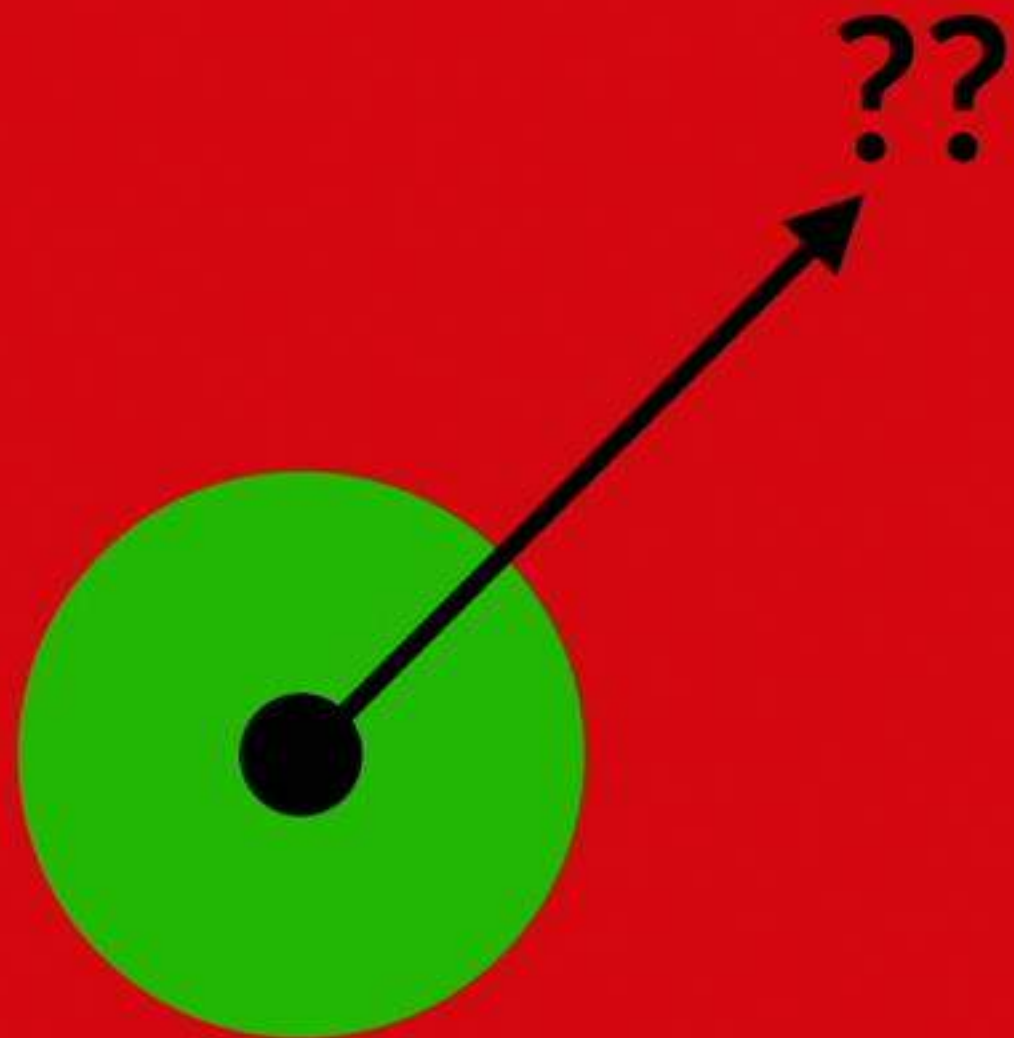
Trust Regions



Trust Regions



Trust Regions



Trust Regions

TRPO and PPO: Motivated by KL-based trust region:

$$\max_s D_{KL} \left(\pi_{\theta_{t+1}}(\cdot | s) \parallel \pi_{\theta_t}(\cdot | s) \right) \leq \delta$$

“keep the max distance between action distributions small”

But relax to an expectation:

$$\mathbb{E}_{s \sim \theta_t} \left[D_{KL} \left(\pi_{\theta_{t+1}}(\cdot | s) \parallel \pi_{\theta_t}(\cdot | s) \right) \right] \leq \delta$$

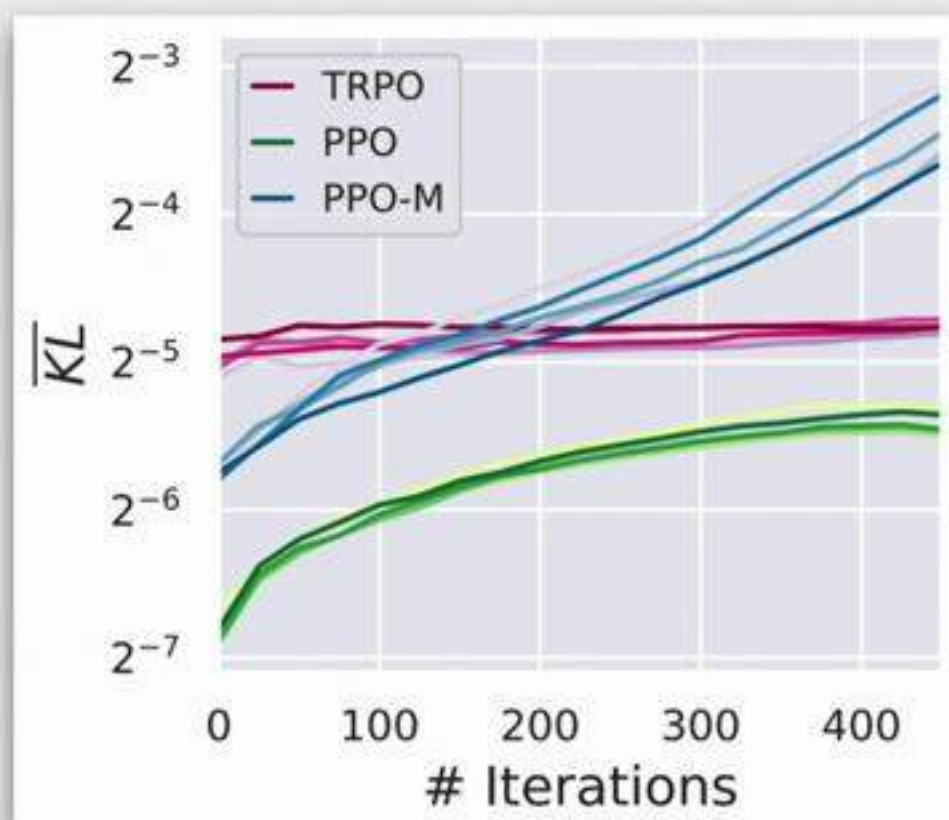
“keep the mean distance between action distributions small”

Trust Regions

What happens in practice?

Trust Regions

Mean KL Distance



- ▶ **TRPO** maintains trust region
- ▶ **PPO** algorithm does not!
- ▶ ... but **optimizations** help

Trust Regions

- ▶ What part of algorithms keep trust regions?
- ▶ How do we reason about algorithms when they use such loose relaxations?
- ▶ How can we capture different kinds of uncertainty in our trust regions?

Takeaways

Recap

- ▶ Deep RL methods are complicated
- ▶ Deep RL training dynamics are poorly understood
 - ▶ Steps are often uncorrelated
 - ▶ Surrogate rewards do not match true rewards
 - ▶ Trust regions do not hold

How do we proceed?

- ▶ **Reconciling RL with our conceptual framework**
 - ▶ How can we make algorithms better follow our conceptual framework?
- ▶ **Rethinking primitives for modern settings**
 - ▶ How do we deal with *high dimensionality*? Algorithm “*optimizations*?” Non-convex function approximators?
- ▶ **Better evaluation for RL systems**
 - ▶ Benchmarks don't capture *reliability, safety, or robustness* of RL agents

Read more

Paper



<https://bit.ly/2UQGpad>

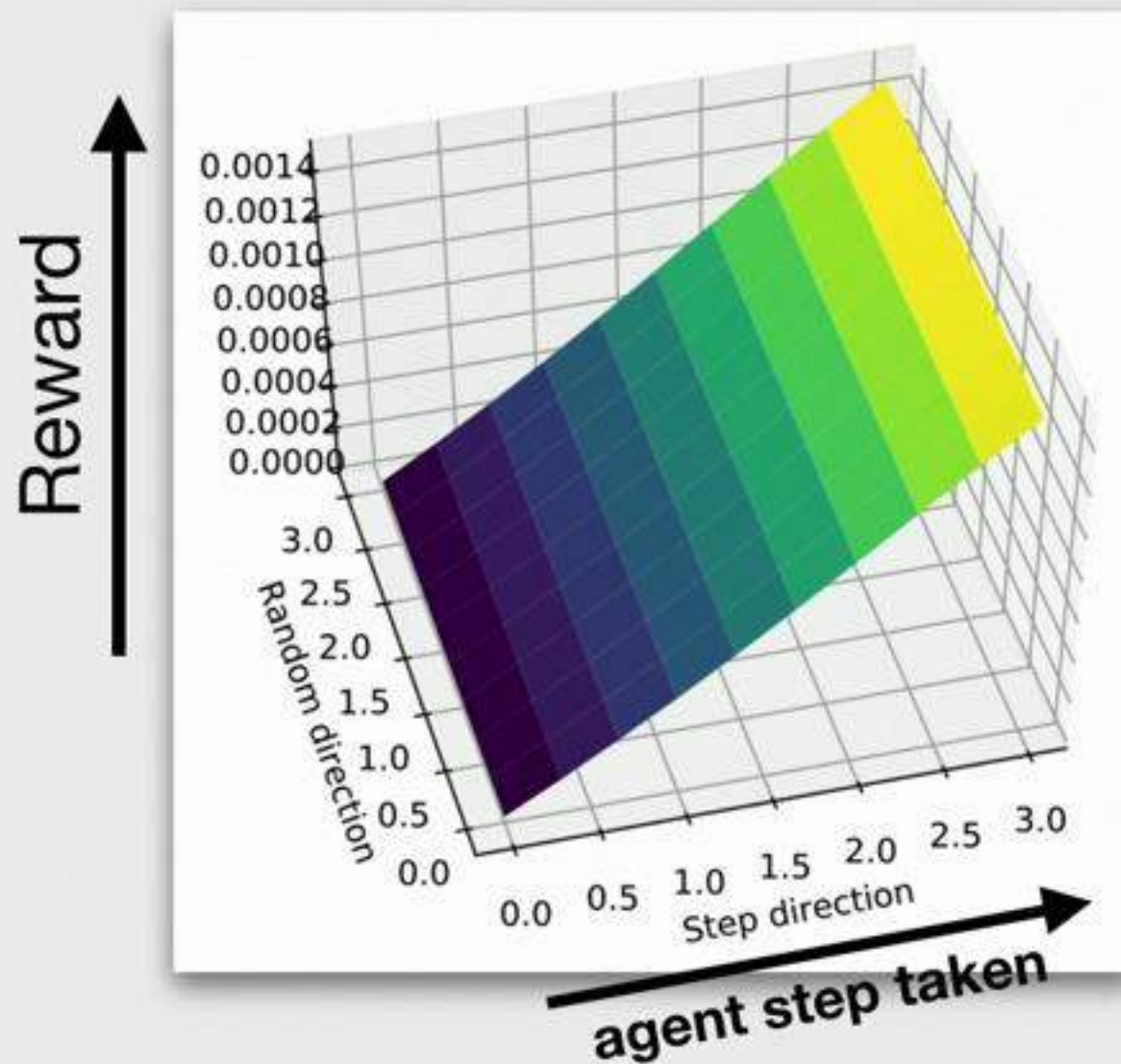
Blog Posts



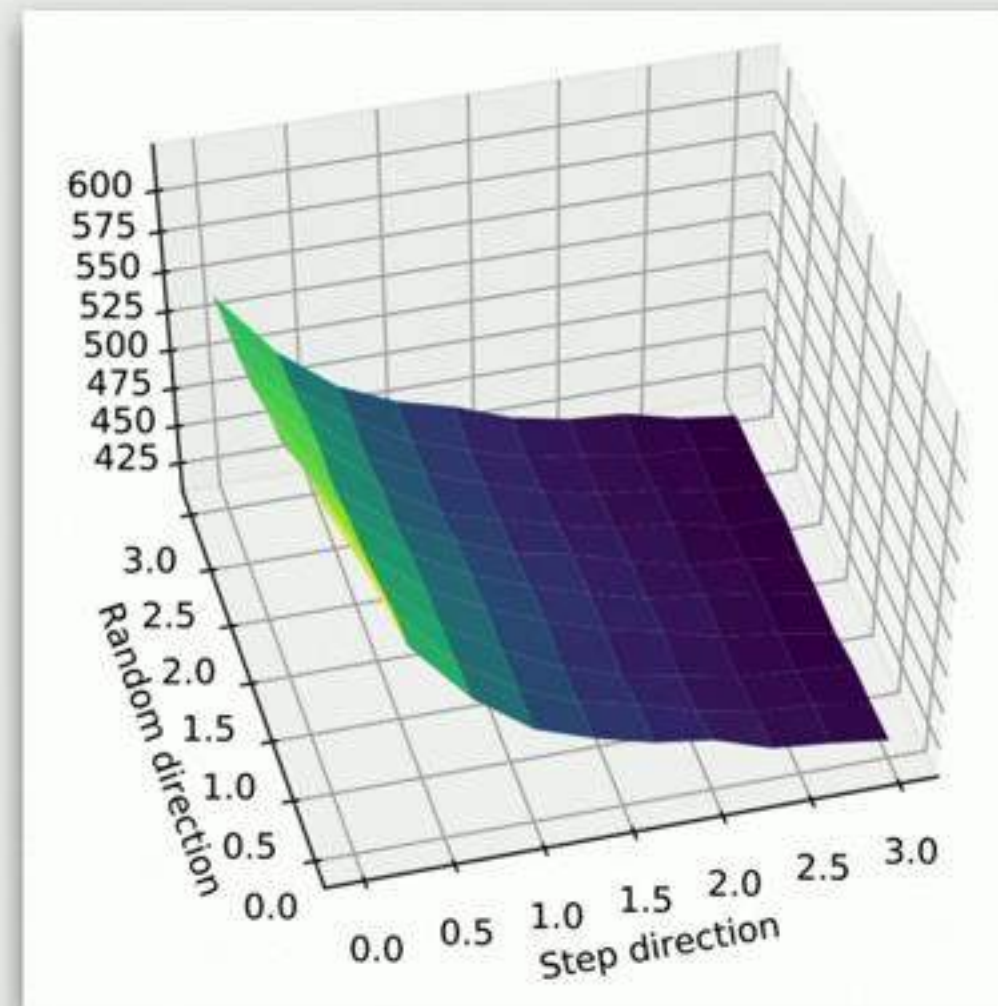
<https://bit.ly/2WVPjAn>

Optimization Landscapes

Surrogate Landscape



Reward Landscape



Step 450

Read more

Paper

Revisiting the Primitives of Deep Policy Gradient Algorithms

Anonymous Authors¹

Abstract

We study how the behavior of deep policy gradient algorithms reflects the underlying framework motivating their development. We present a fine-grained analysis of state-of-the-art methods based on key aspects of this framework: gradient estimation, value prediction, optimization landscapes, and trust region estimators. From this perspective, the behavior of deep policy gradient algorithms often deviates from what their motivating framework may predict. Our analysis suggests first steps towards solidifying the foundations of these algorithms, and in particular indicates that we may need to move beyond the current benchmark-centric evaluation methodology.

1. Introduction

Deep reinforcement learning (RL) is at the core of some of the most published achievements of modern machine learning (Silver et al., 2017; OpenAI, 2018; Depreux et al., 2018). To many, this framework embodies the promise of the real-world impact of machine learning. However, the deep RL toolkit has not yet attained the same level of engineering maturity as, for example, the current deep supervised learning framework. Indeed, recent studies (Fedorov et al., 2017) demonstrate that state-of-the-art deep RL algorithms suffer from inconsistency in hyperparameters, lack of consistency, and poor reproducibility.

This state of affairs suggests that it might be necessary to re-evaluate the conceptual underpinnings of deep RL methodology. More precisely, the overarching question that motivates this work is:

To what degree does the current practice of deep RL reflect the principles that informed its development?

The specific focus of this paper is on deep policy gradient methods, a widely used class of deep RL algorithms.

¹Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author email@domain.com.

²Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Our goal is to explore the extent to which state-of-the-art implementations of these methods succeed at realizing the key primitives of the general policy gradient framework.

1.1. Our contributions

Our point of start is a prominent deep policy gradient method: general policy optimization (GPO) (Schulman et al., 2017). We find that GPO’s performance depends heavily on optimizations outside of the core algorithm. This suggests that the practical success of GPO might not be fully explainable by its motivating theoretical framework.

This observation prompts us to take a broader look at policy gradient algorithms and their relation to their underlying framework. With this perspective in mind, we perform a fine-grained examination of key RL primitives as they manifest in practice. Concretely, we study:

Gradient Estimation: we find that even while agents are improving in terms of reward, the gradient estimates used to update their parameters are often poorly correlated with the true gradient². We also find that gradient estimate quality decays with training progress and task complexity.

Value Prediction: our experiments indicate that value networks successfully solve the supervised learning task they are trained on, but do not fit the true value function. Additionally, employing a value network as a baseline only marginally decreases the variance of gradient estimates compared to using raw values (our dynamically increasing agent’s performance compared to using no baseline at all).

Optimization Landscapes: we also observe that the optimization landscape induced by modern policy gradient algorithms is often not reflective of the underlying true reward landscape, and that the latter is often poorly behaved in the relevant sample region.

Trust Regions: we find deep policy gradient algorithms sometimes violate theoretically motivated trust regions. In fact, in proximal policy optimization, these violations stem from a fundamental problem in the algorithm’s design.

²How this does not precisely gradients from being useful. Indeed, this is known to be the case in many machine learning settings. However, it is unclear to what degree we can trust iterations from these strings in deep RL. Also, we find that gradient variance interacts in complex ways with other factors.

Blog Posts



<https://bit.ly/2WVPjAn>

<https://bit.ly/2UQGpad>