

Reliable RL: An Algorithmic Perspective

Logan Engstrom

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



gradsci.org



CSAIL

Reinforcement Learning (RL)



Reinforcement Learning (RL)



Reinforcement Learning (RL)

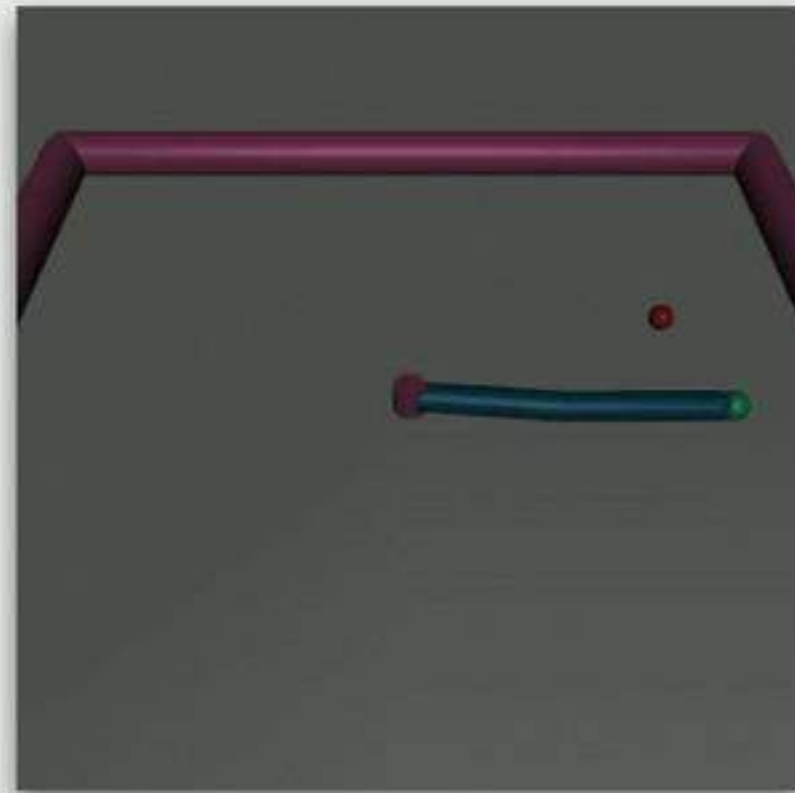
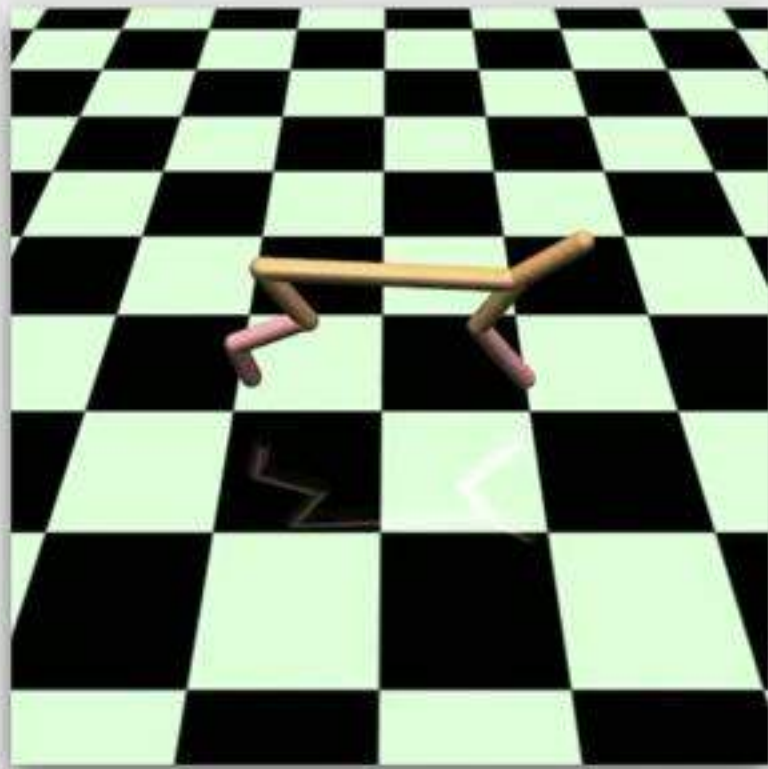


Is RL “real-world-ready”?

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

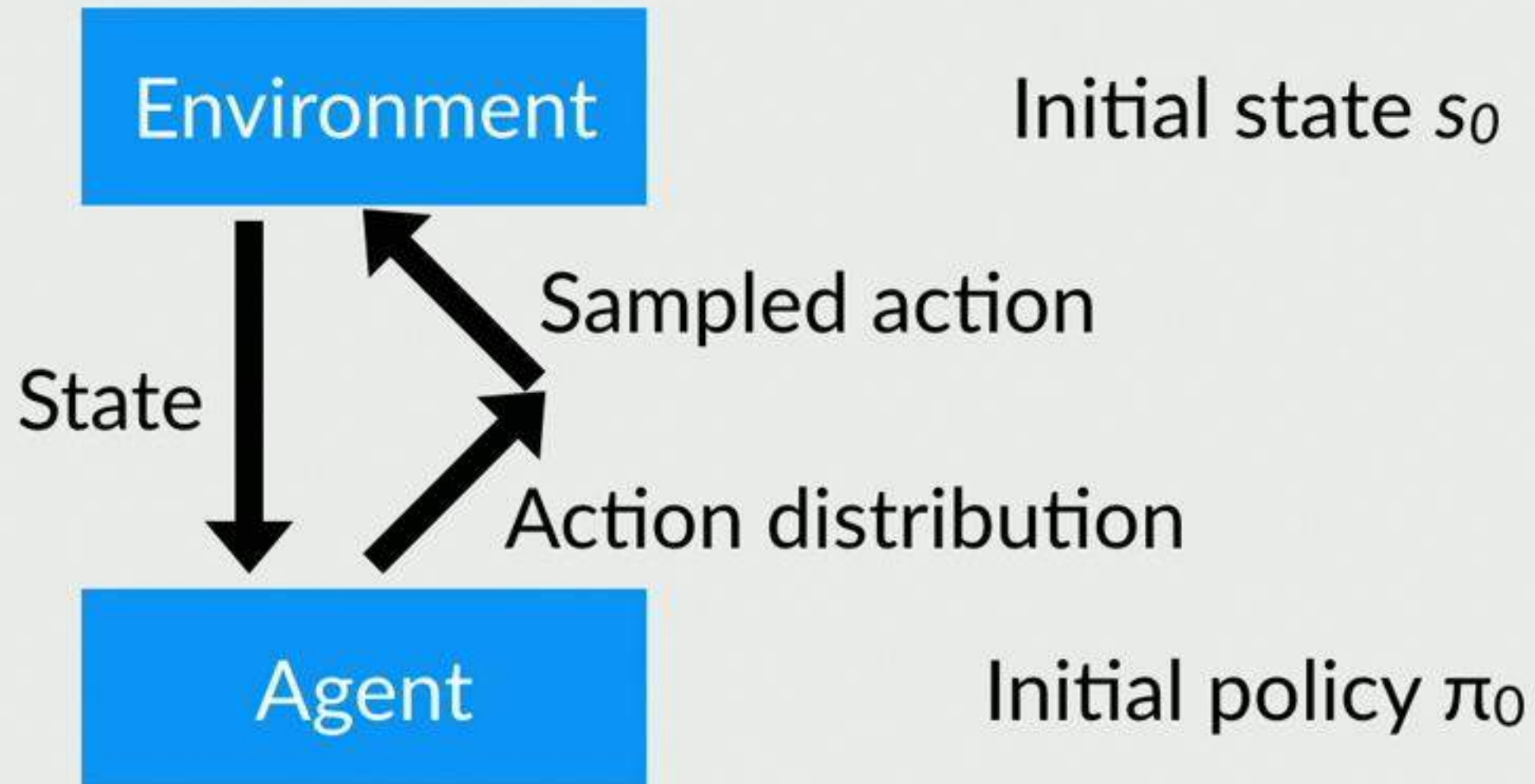
Environment

Initial state s_0

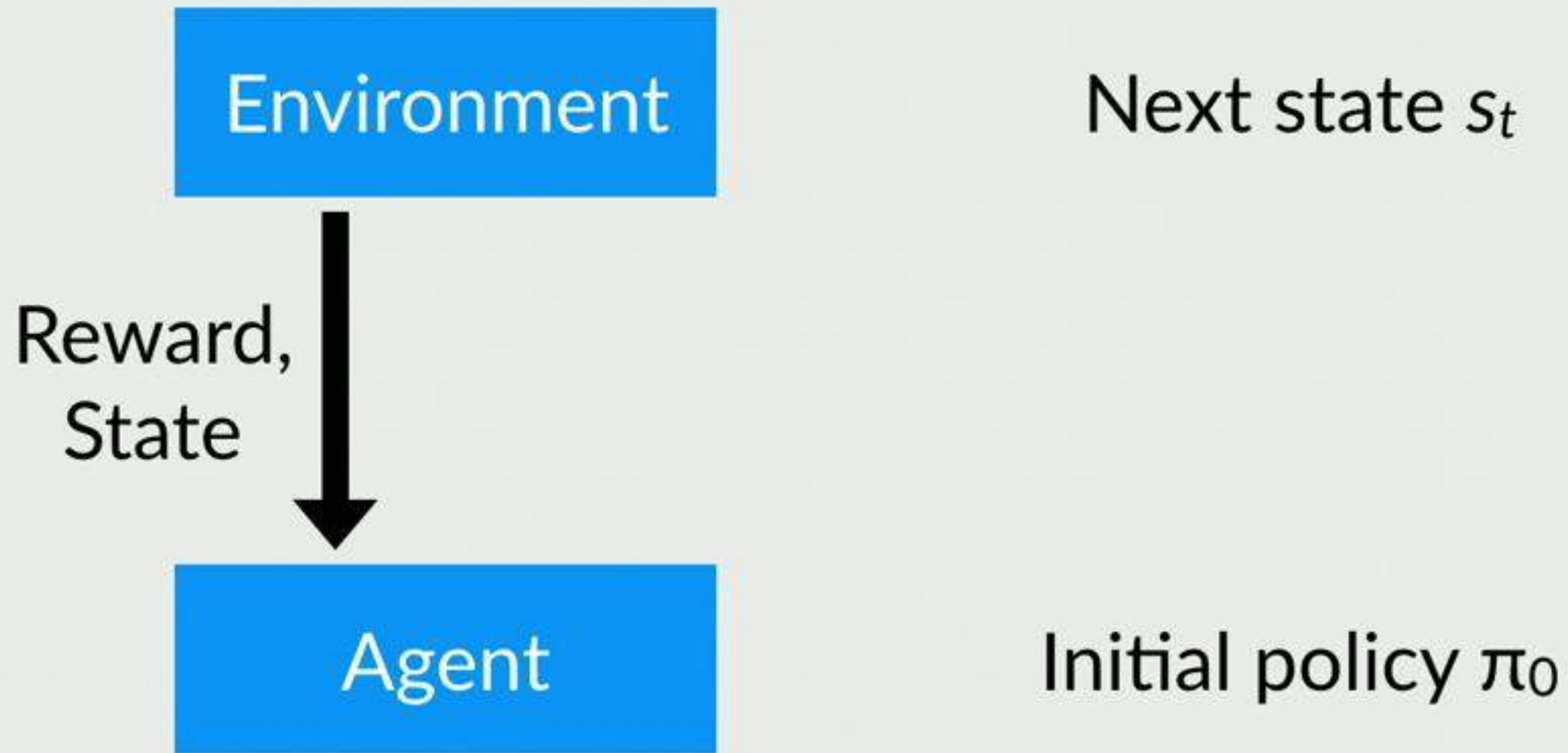
Agent

Initial policy π_0

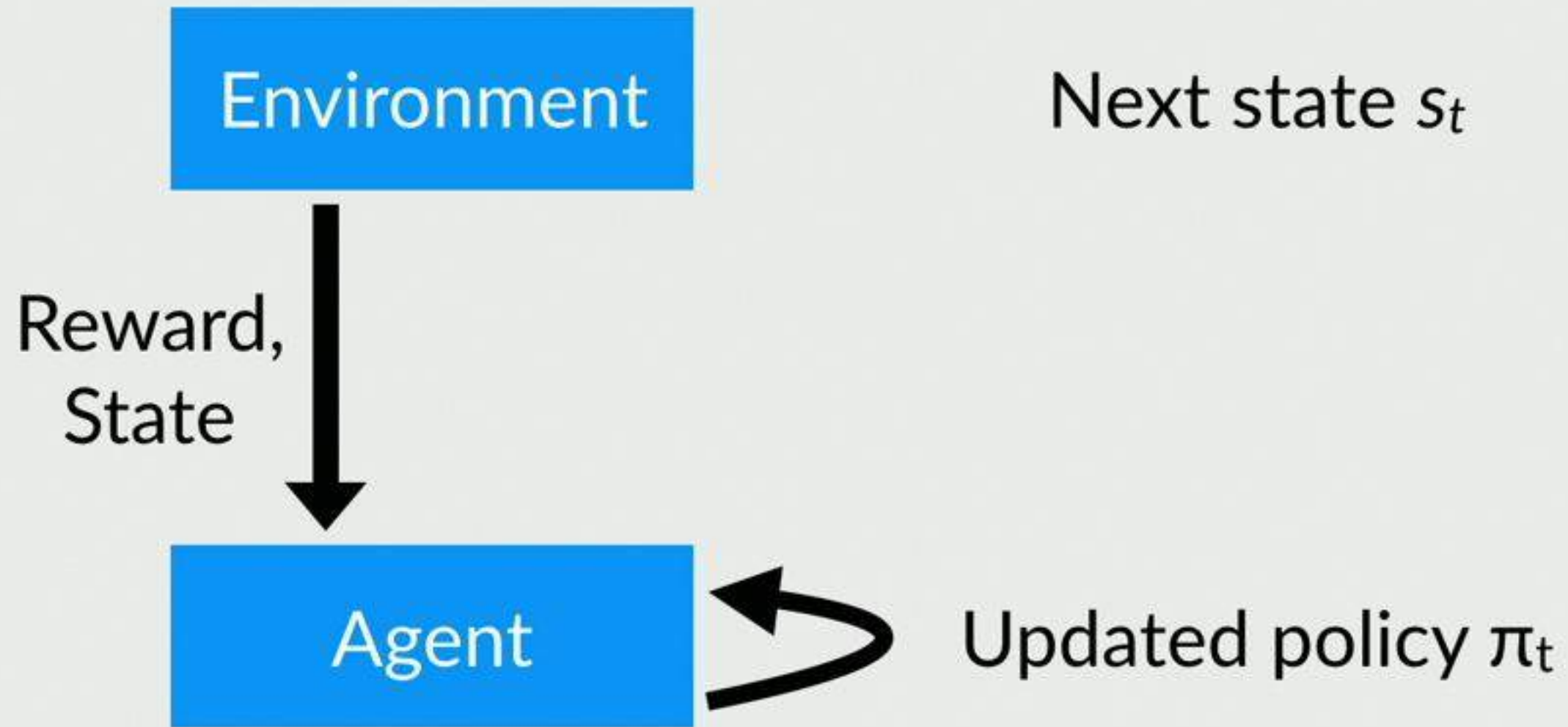
The RL Setup



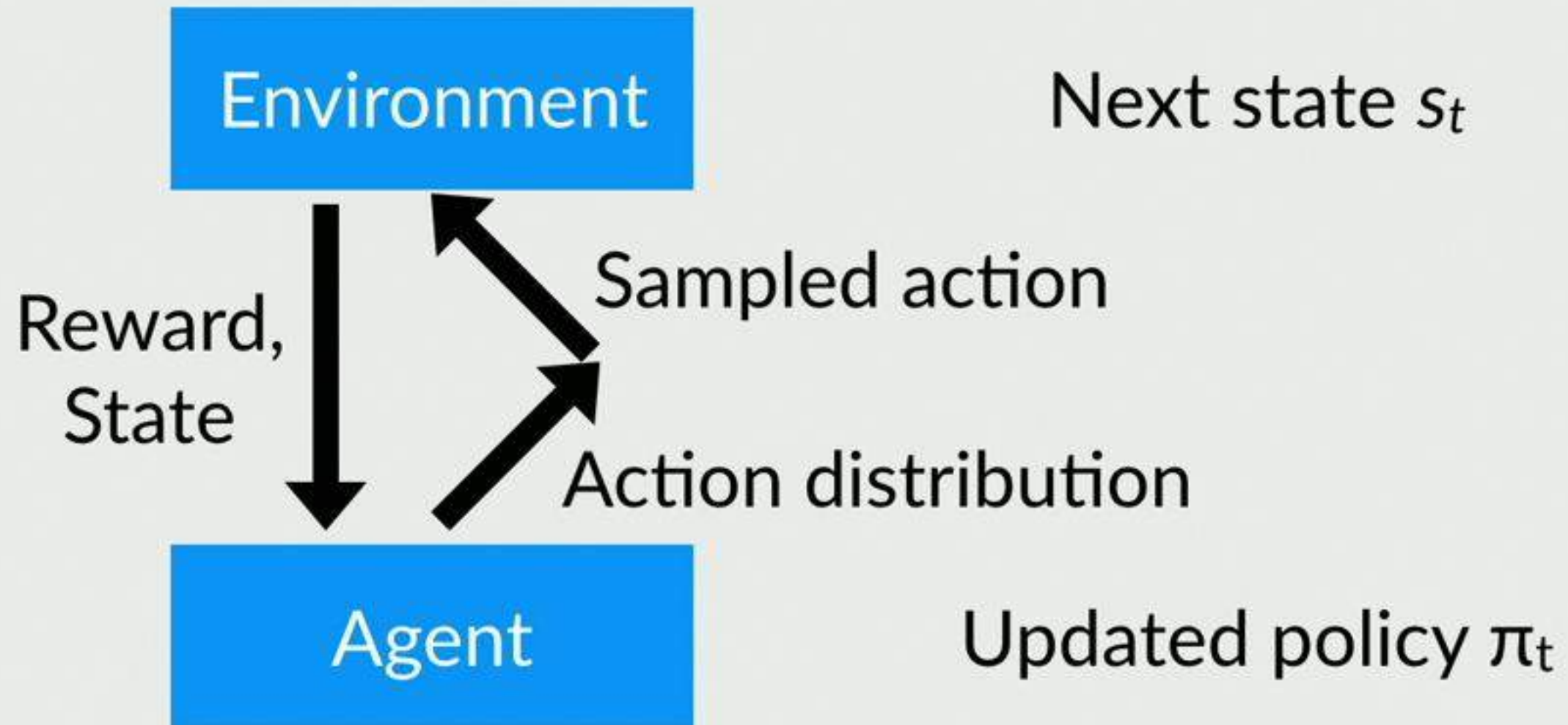
The RL Setup



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

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

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

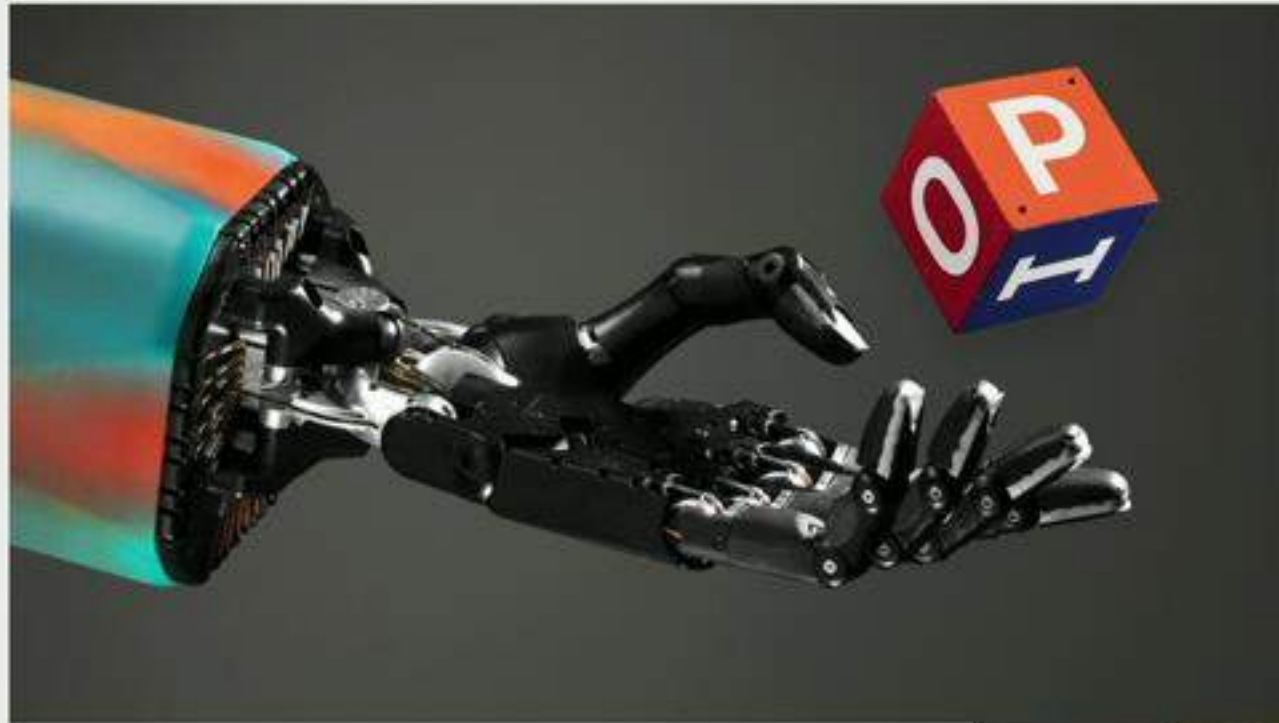
Can we instead get a good *estimate* of the gradient?

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

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

Then: use estimate in gradient descent!

Policy Gradient Successes



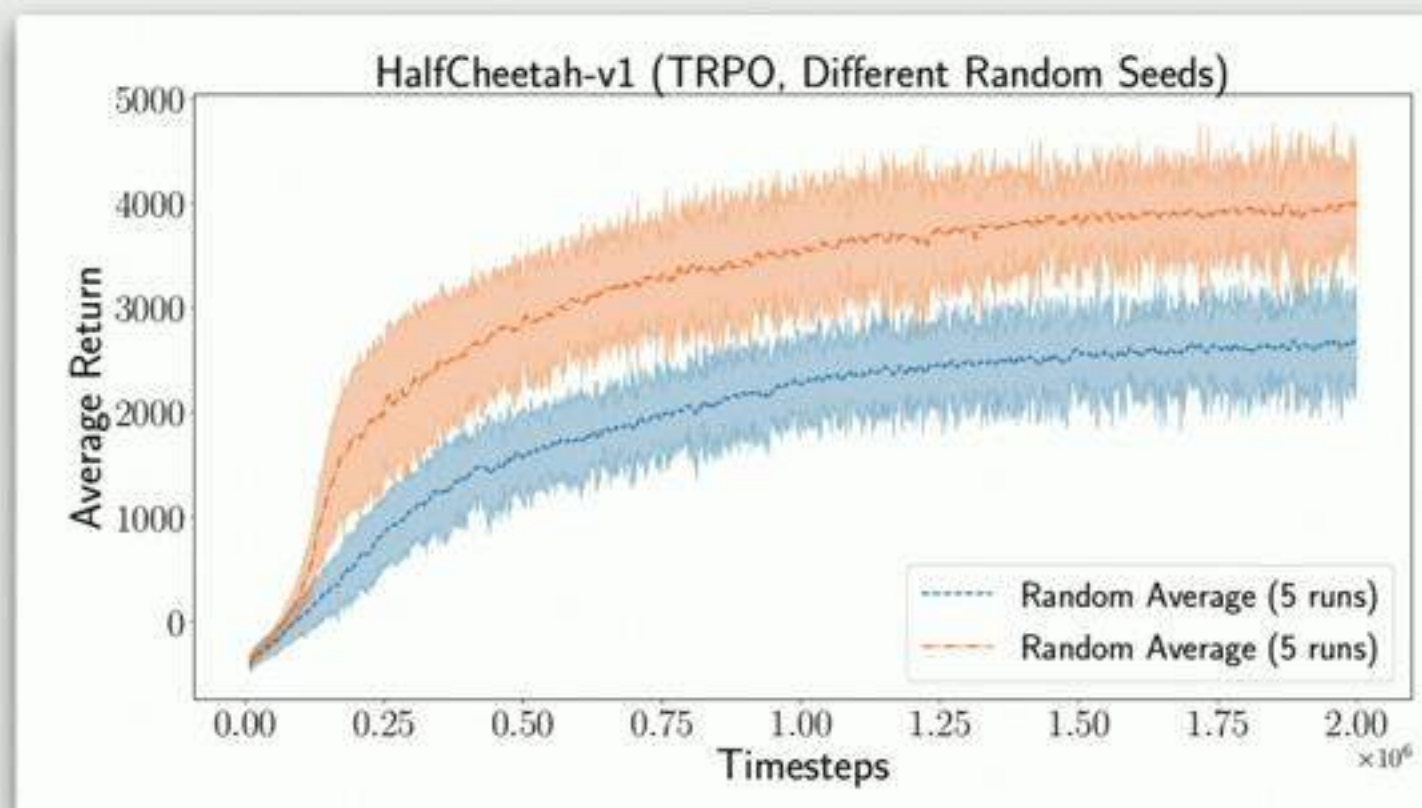
The Rotten Truth of Deep RL



The Rotten Truth of Deep RL

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

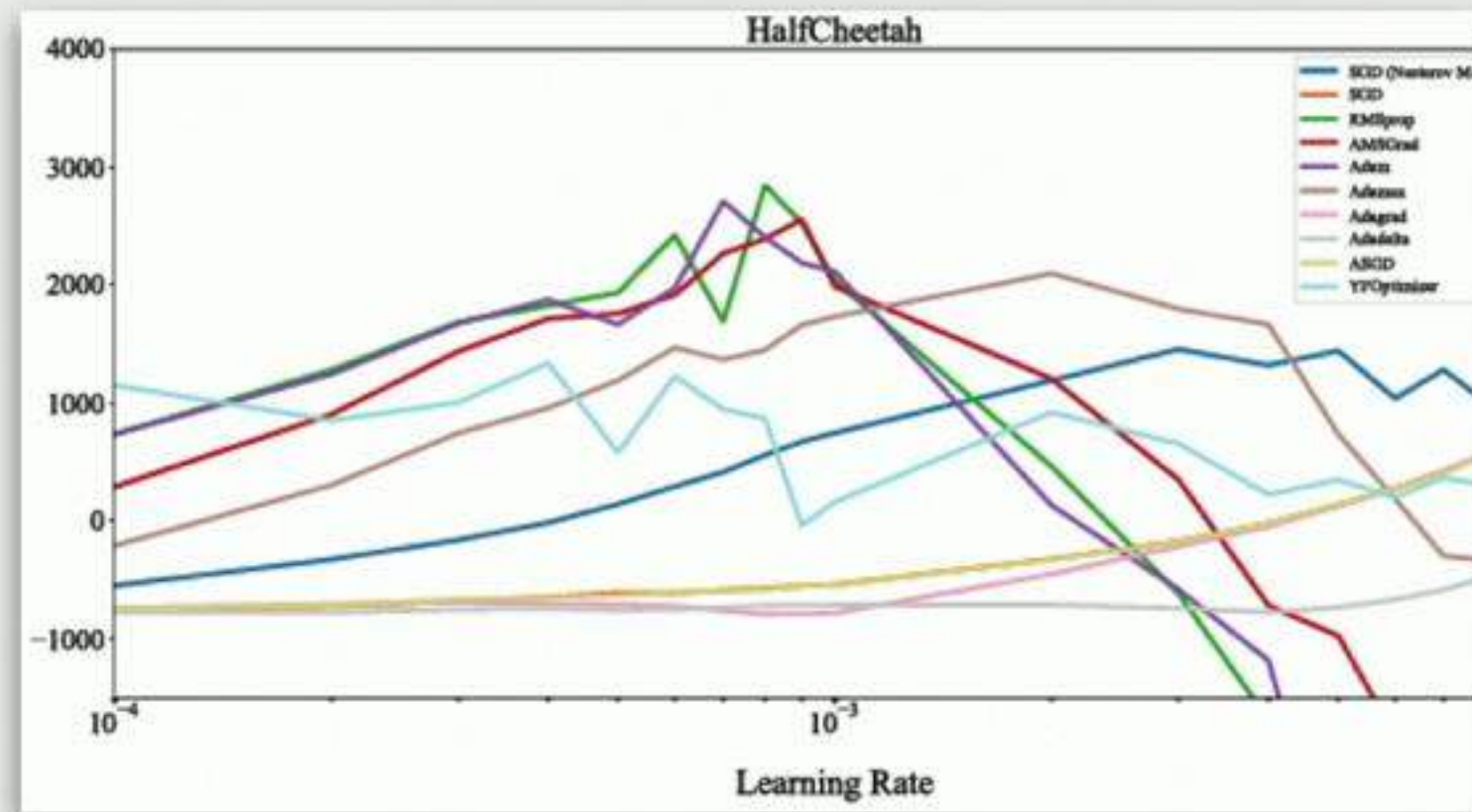
- ▶ Poor reliability over repeated runs



The Rotten Truth of Deep RL

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

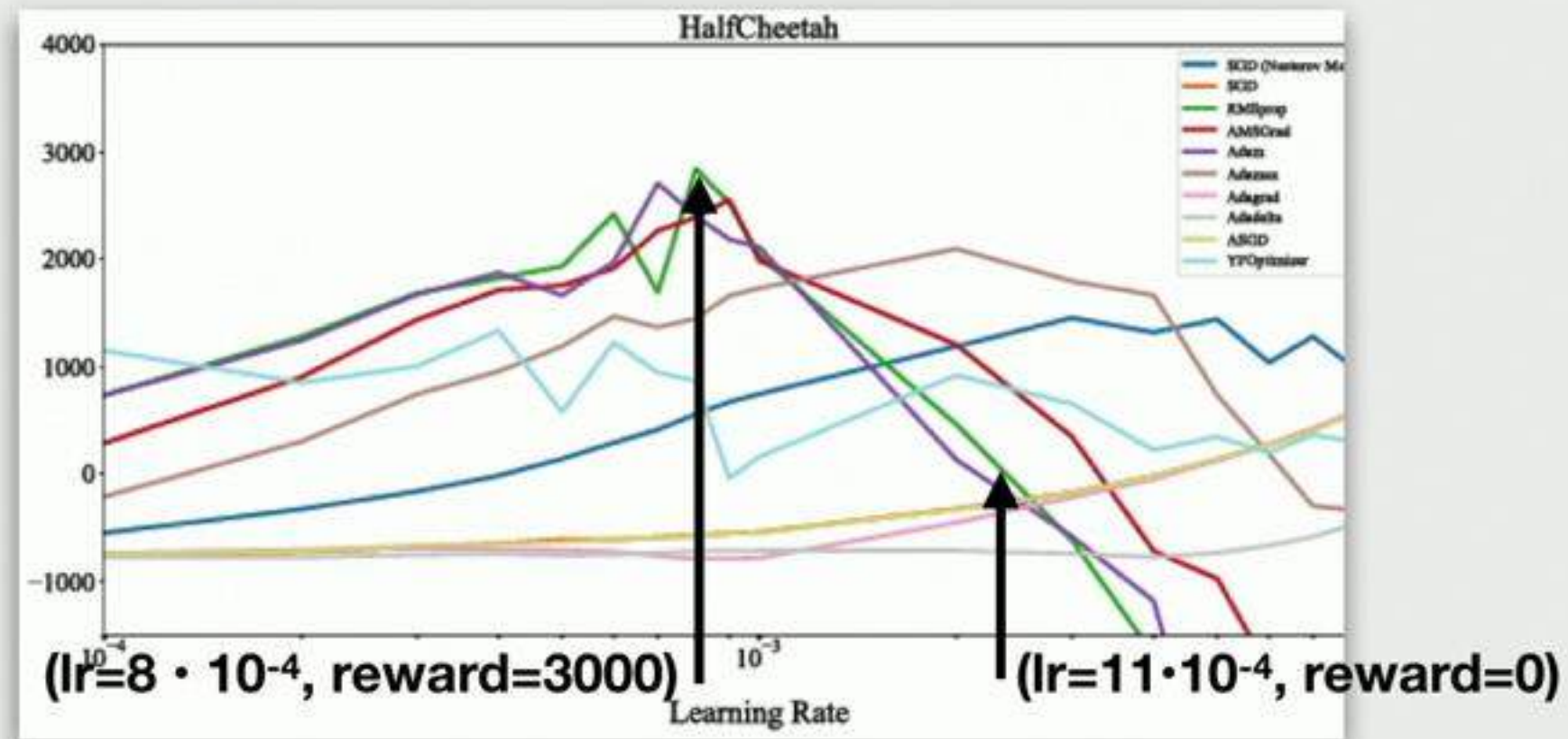
- ▶ Poor reliability over repeated runs
- ▶ High sensitivity to hyperparameters



The Rotten Truth of Deep RL

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

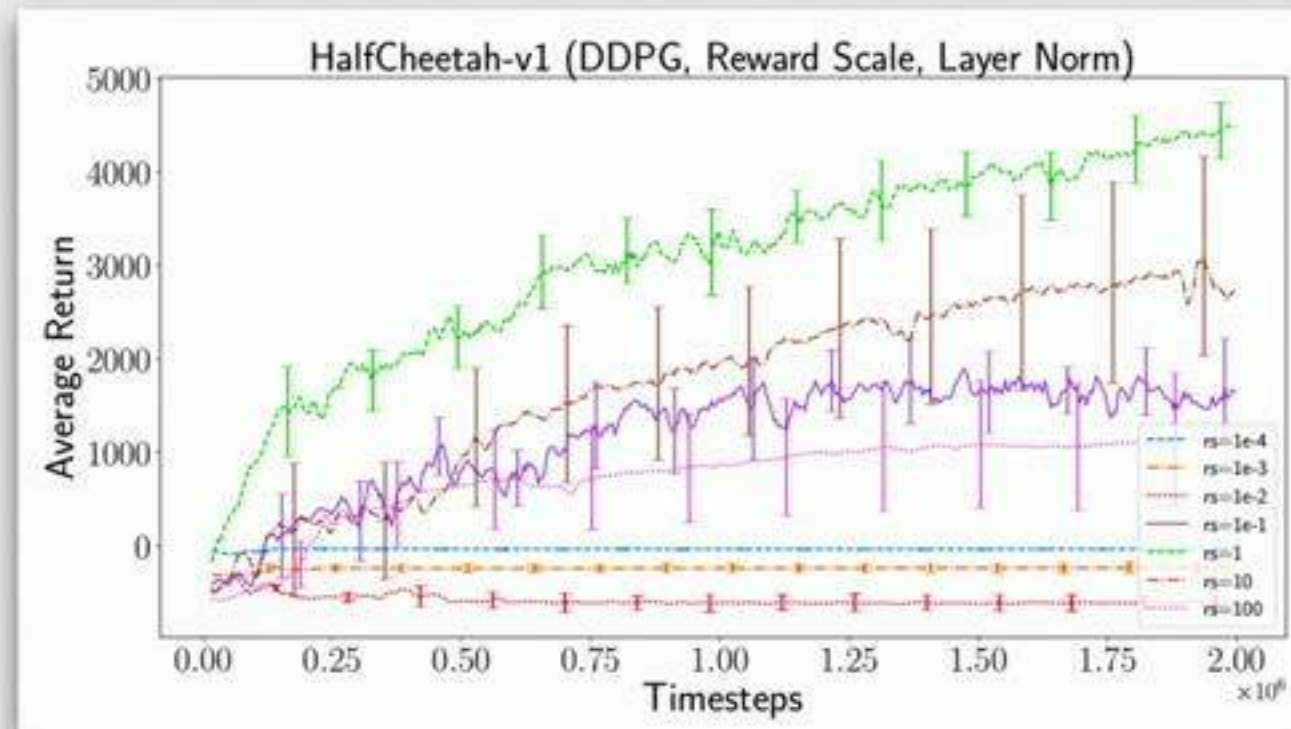
- ▶ Poor reliability over repeated runs
- ▶ High sensitivity to hyperparameters



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

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 open, the `MapPolicy` setup has an important modification and honestly, I don't understand why it's such a modification.
- Huge architectural differences. In open you get an implementation of the `Rollout` class.

Source: GitHub issues

Implementation Obscures Deep RL Algorithms



- Differences between the policies. In `ppo`, the `klPolicy` setup is an important modification and honestly, I don't understand why it's there.

- Huge architectural differences. It appears you are an implementation of the `Actor-Critic` class.

- Nontrivial changes to the paper, part 2. The code is sprinkled with small tricks. For example, I normalized the advantage function, `adv`. 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`, it's a clipped ratio of action probabilities. `ppo` does that. `ppo` 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 Normalizer class.

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

apthelaw commented 24 days ago

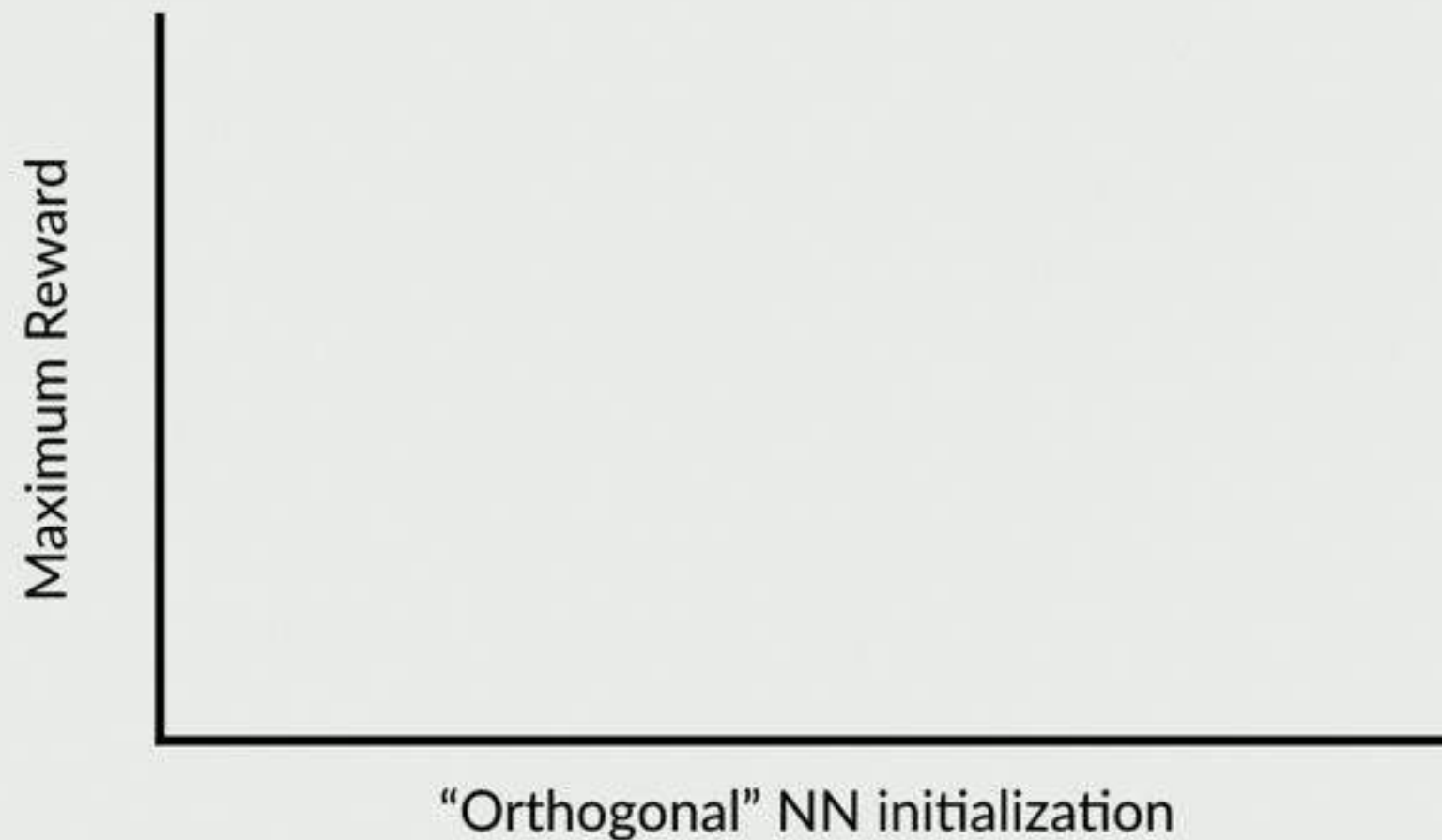
There is one thing between PPO1 and PPO2 that I don't understand. The model actually updates both the old and new set of parameters, a

Nontrivial changes to the paper. I read the openai blogpost and the open paper. Learning at a clipped rate of action probabilities. open blog that ppo1 also use the value function. I would consider that a big difference between ppo and ppo

Source: GitHub issues

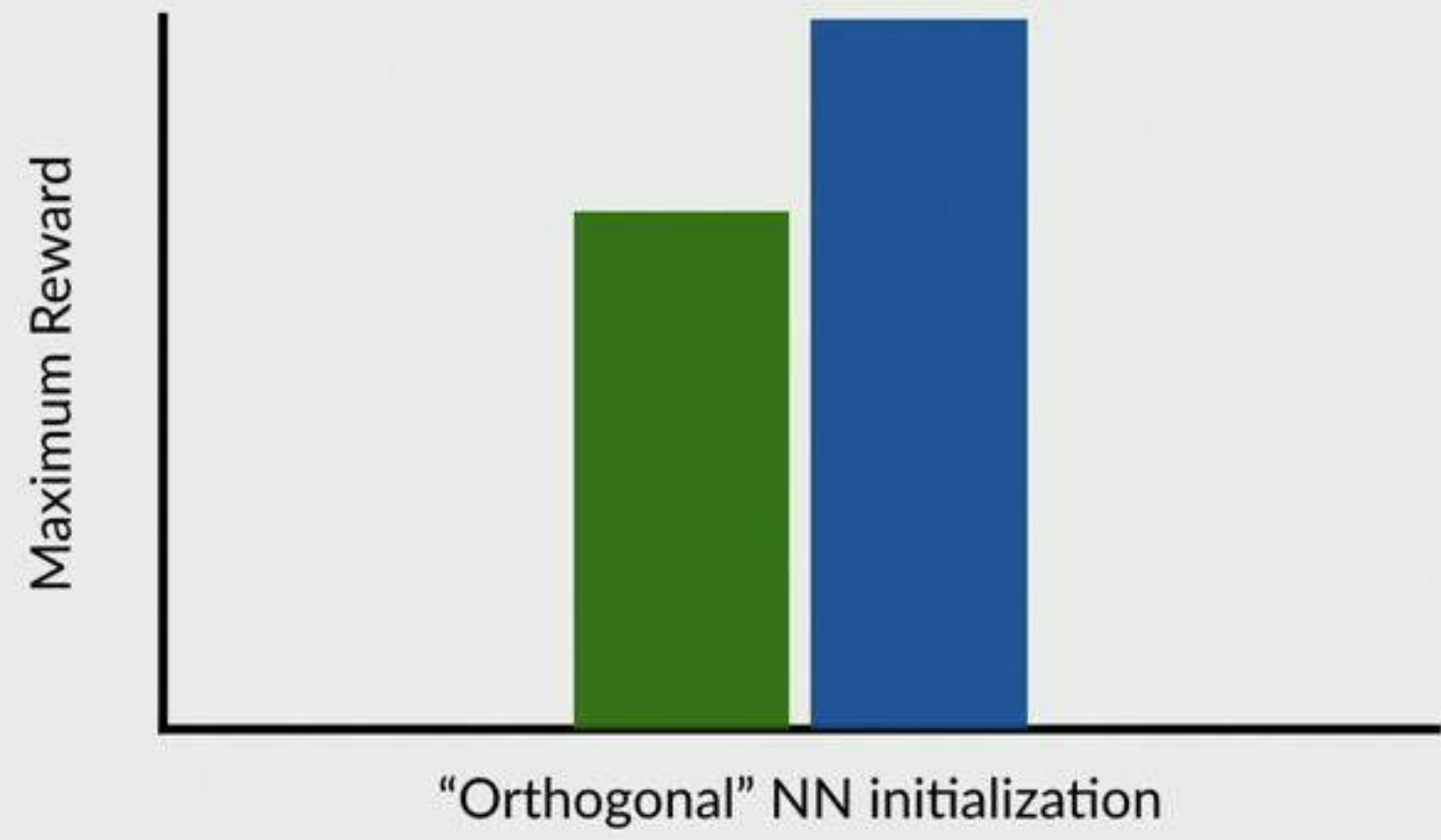
Implementation Obscures Deep RL Algorithms

■ Without Optimization ■ With Optimization

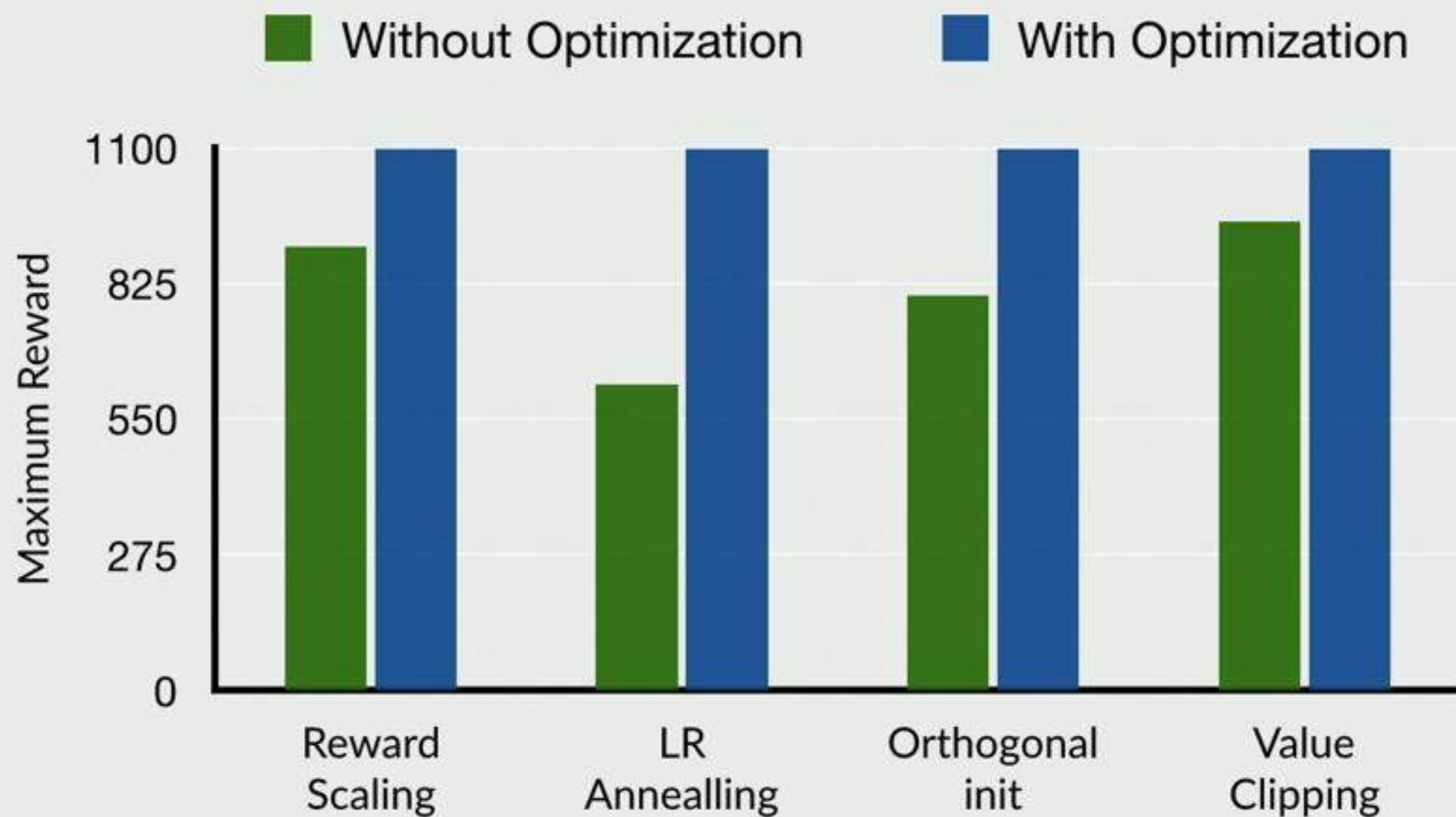


Implementation Obscures Deep RL Algorithms

■ Without Optimization ■ With Optimization



Implementation Obscures Deep RL Algorithms



Implementation Obscures

Deep Dive into the Details

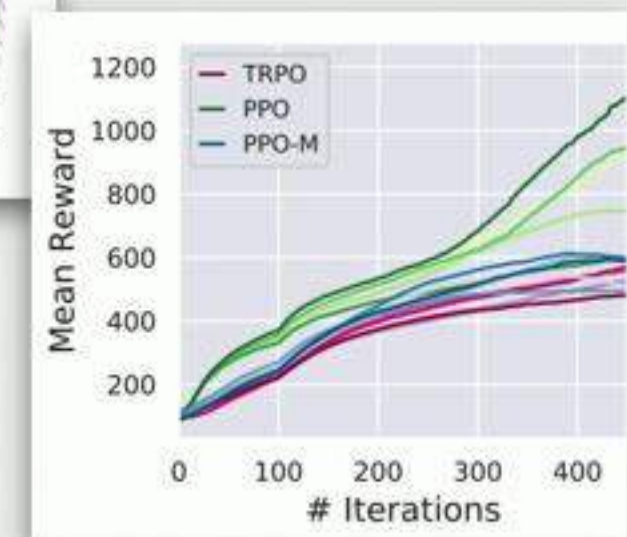
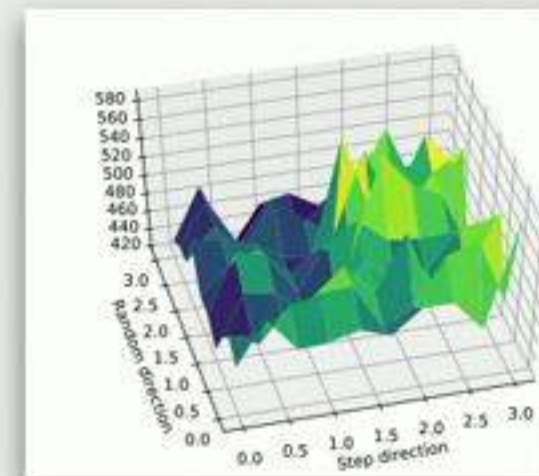
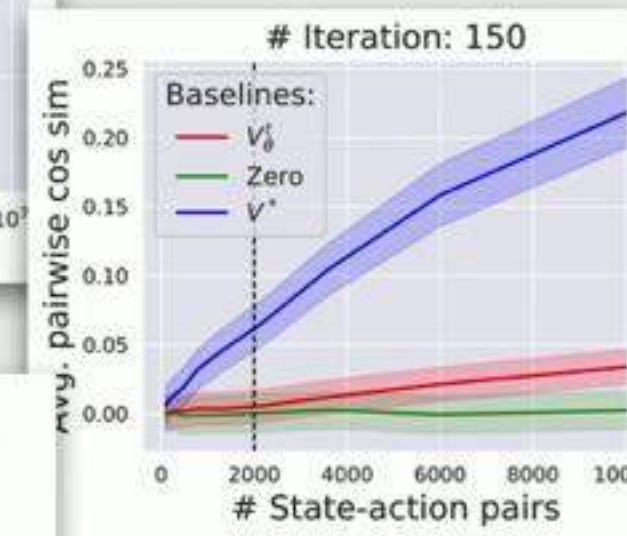
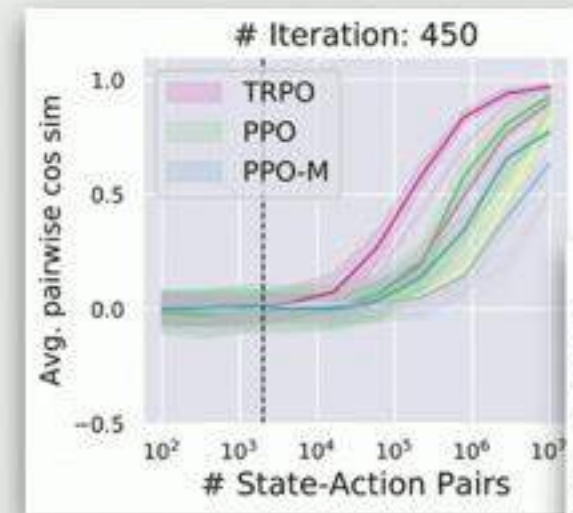
Implementation Obscures Deep RL Algorithms

- ▶ 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:

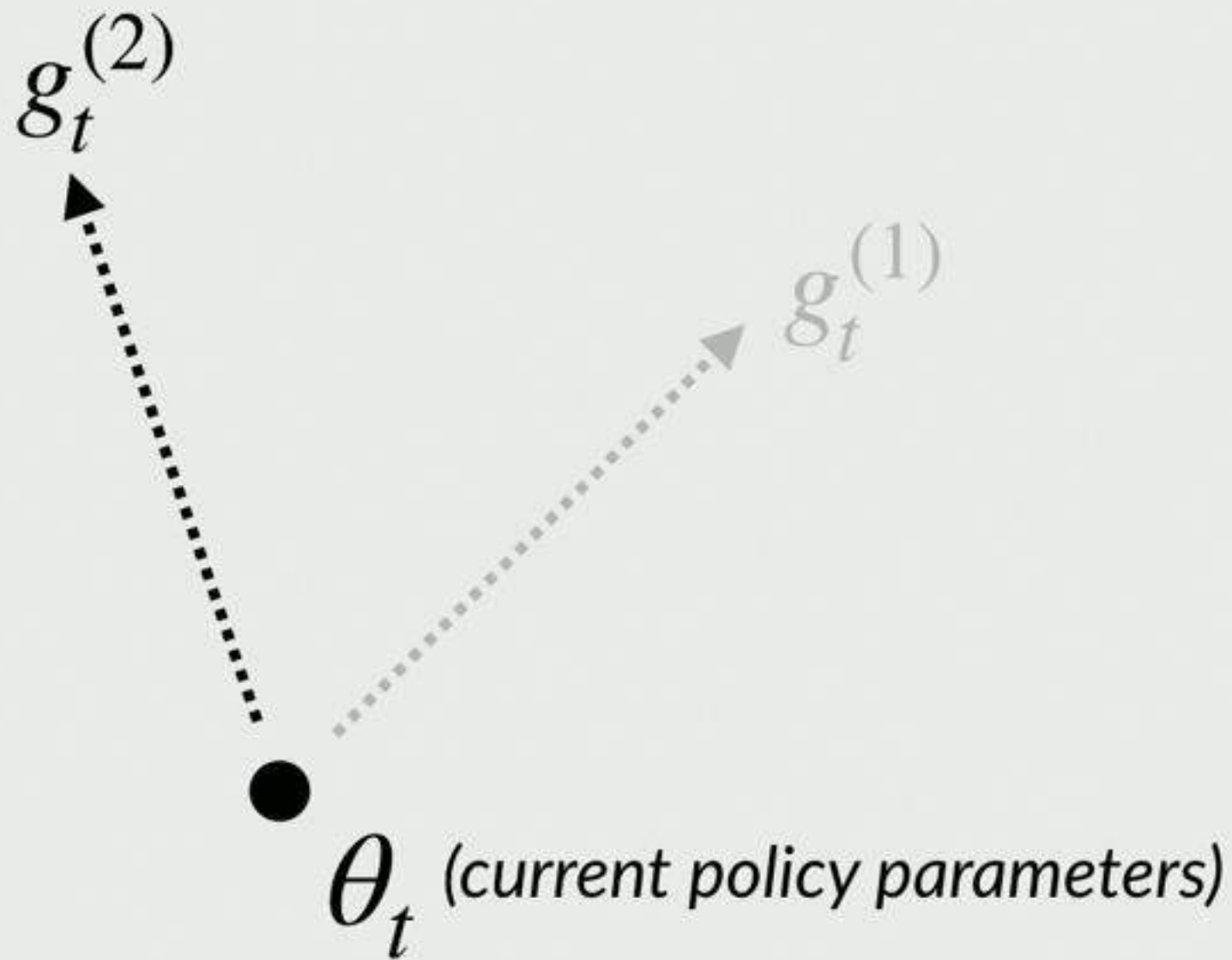
$$\nabla_{\theta} \mathbb{E}_{\tau \sim \theta} [R(\theta)] \approx \frac{1}{N} \sum_{\tau \sim \theta} g(\tau)$$

How valid is this?

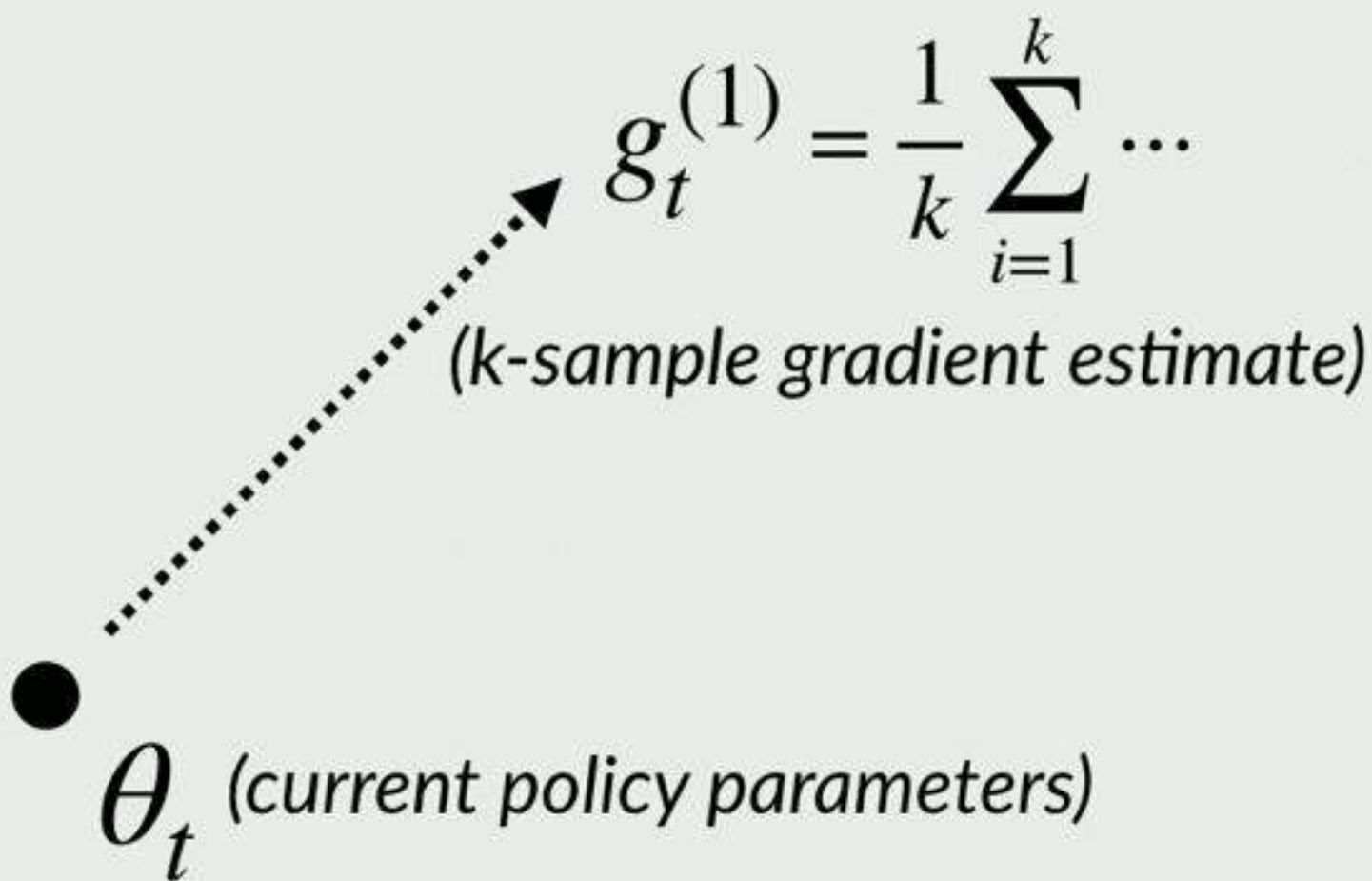
Gradient Estimation

- θ_t (current policy parameters)

Gradient Estimation



Gradient Estimation



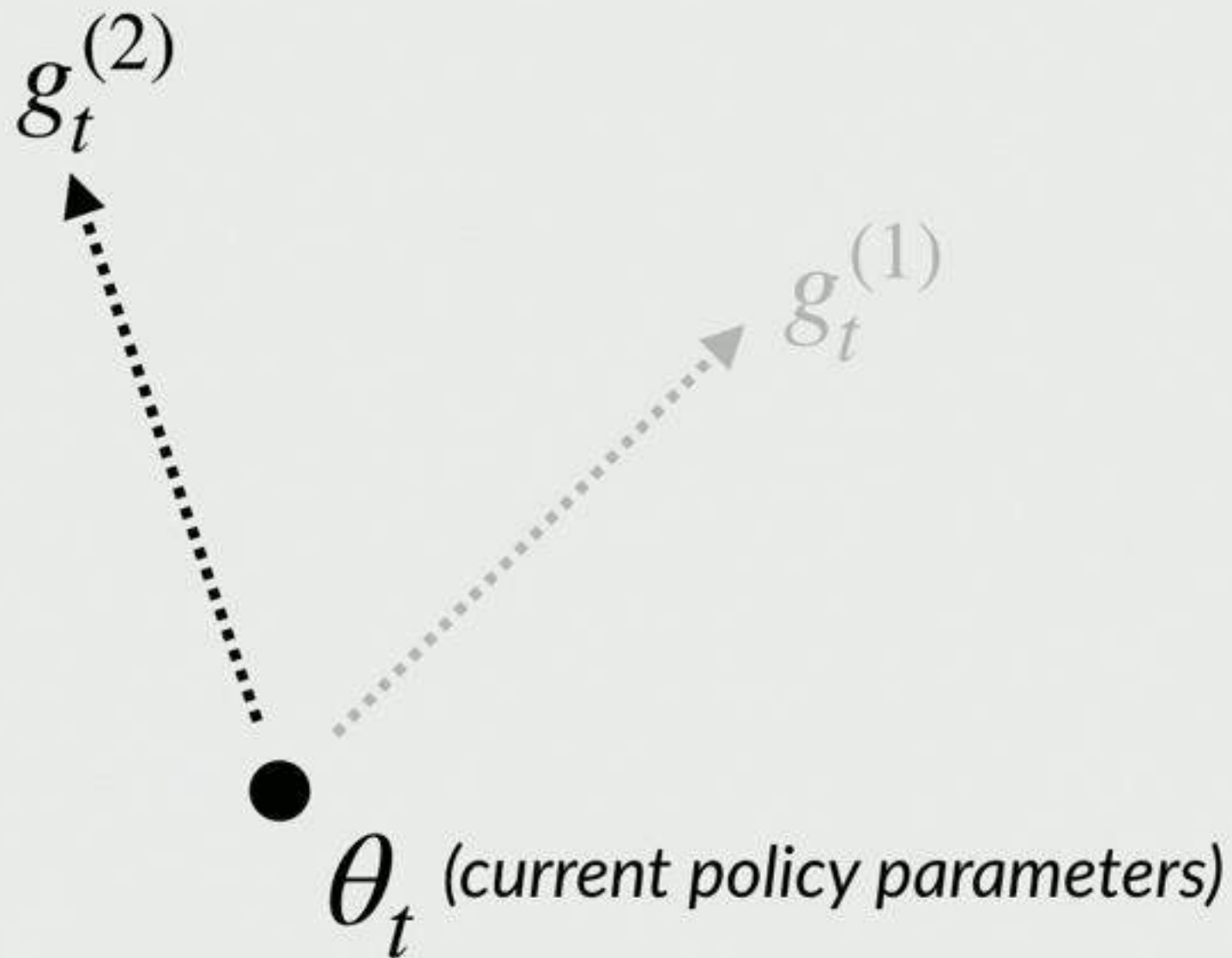
A diagram illustrating gradient estimation. A solid black dot at the bottom left represents the current policy parameters θ_t . A dotted arrow points from this dot towards the upper right, representing the direction of the gradient estimate $g_t^{(1)}$. The equation $g_t^{(1)} = \frac{1}{k} \sum_{i=1}^k \dots$ is written next to the arrow, and the text "(k-sample gradient estimate)" is written below it.

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

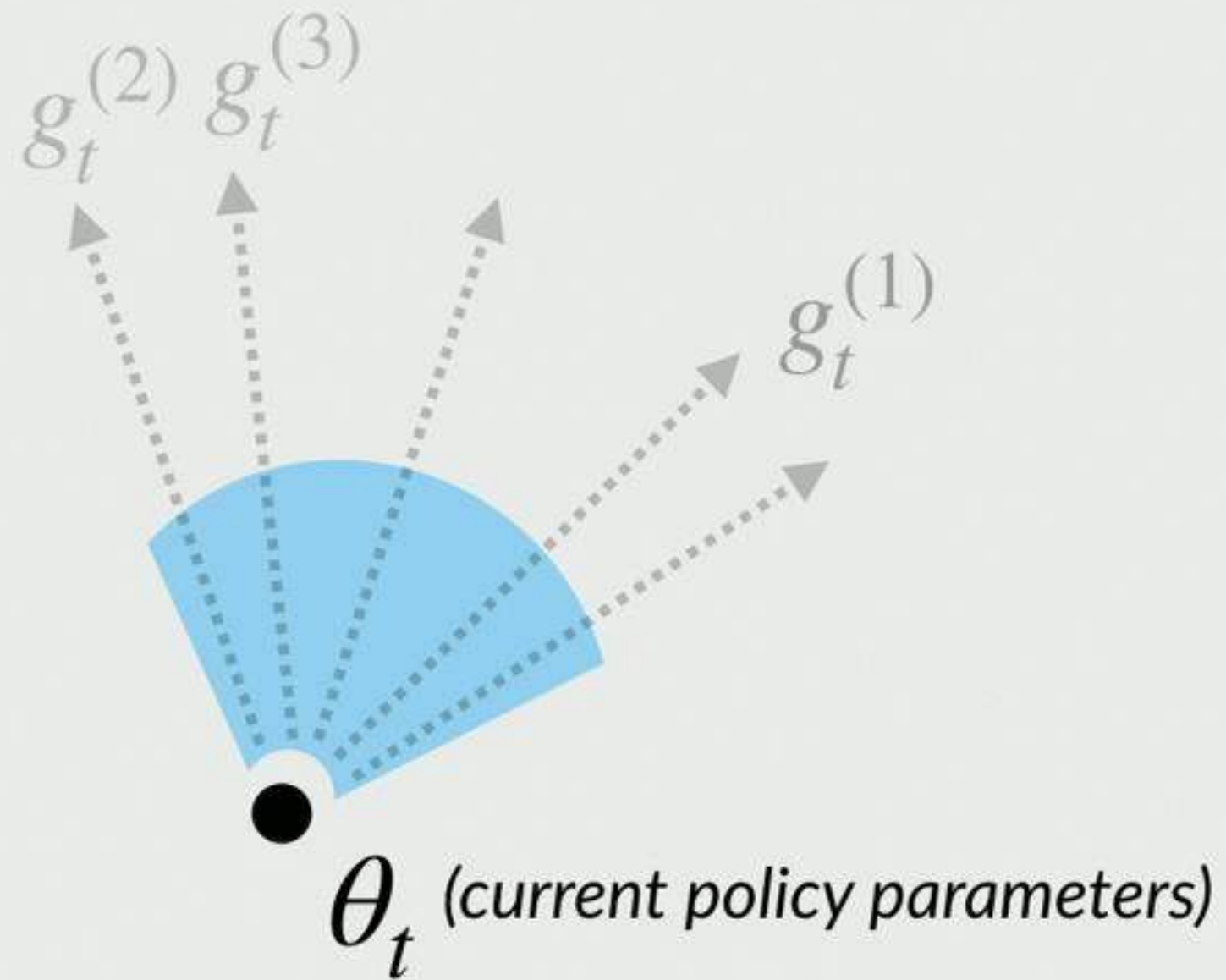
(k-sample gradient estimate)

θ_t (current policy parameters)

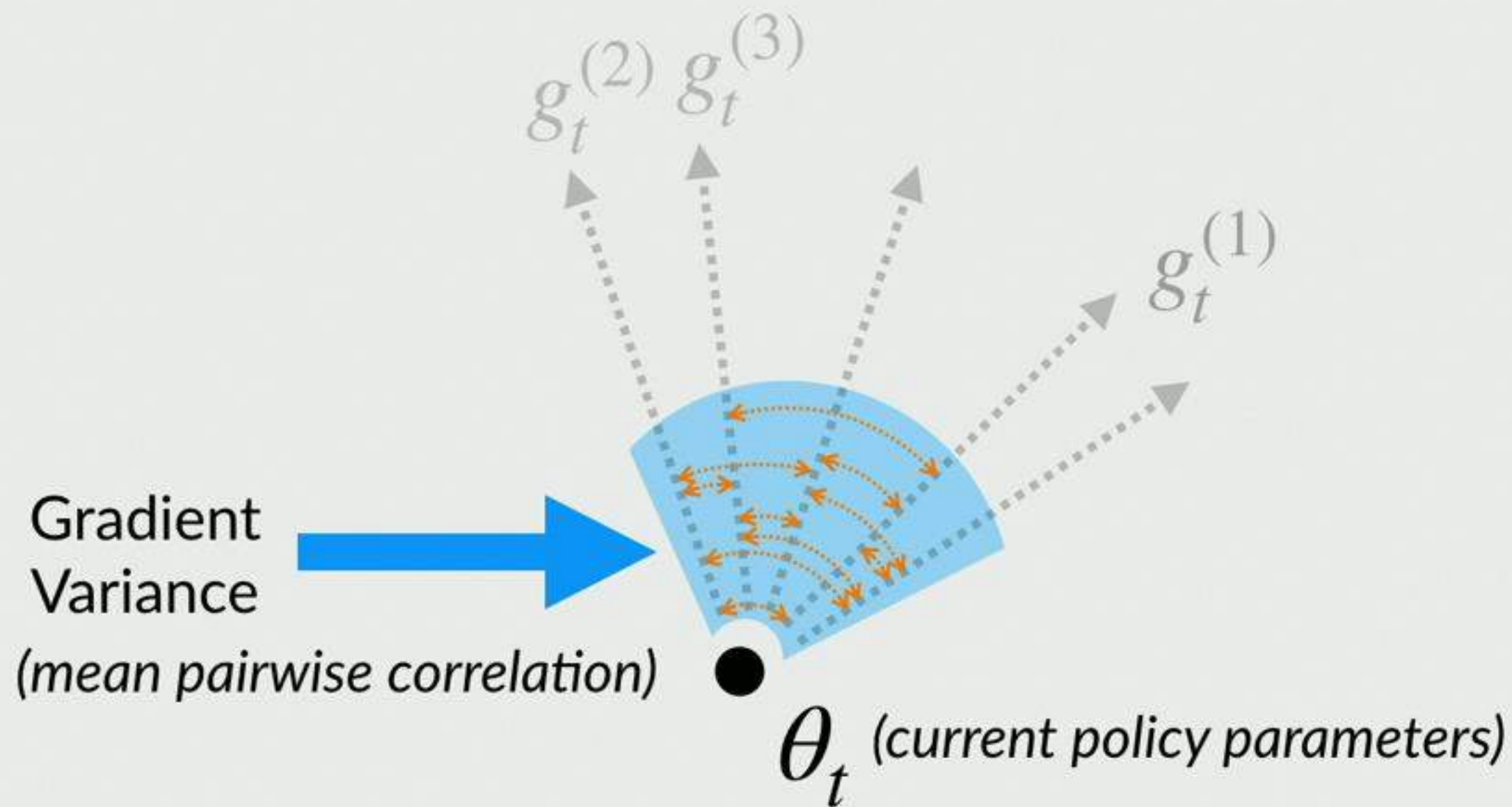
Gradient Estimation



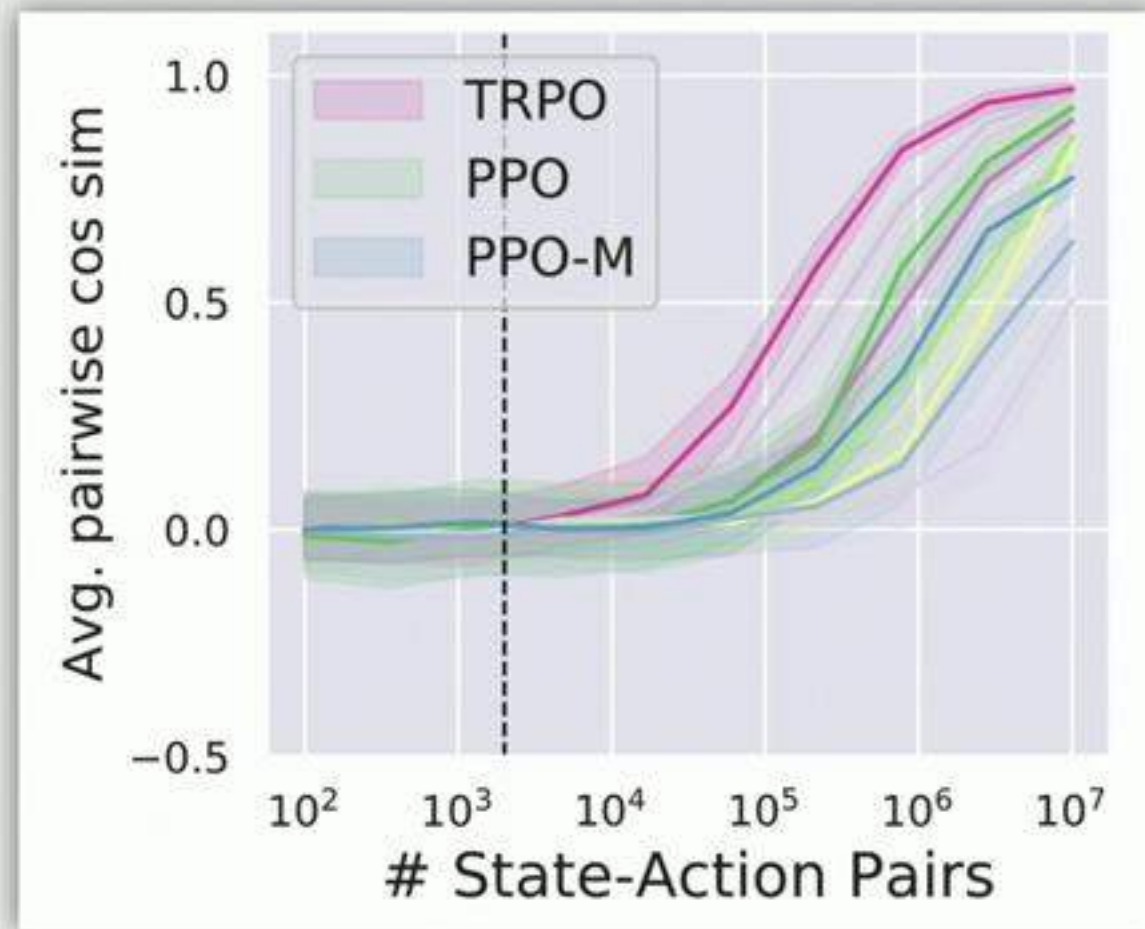
Gradient Estimation



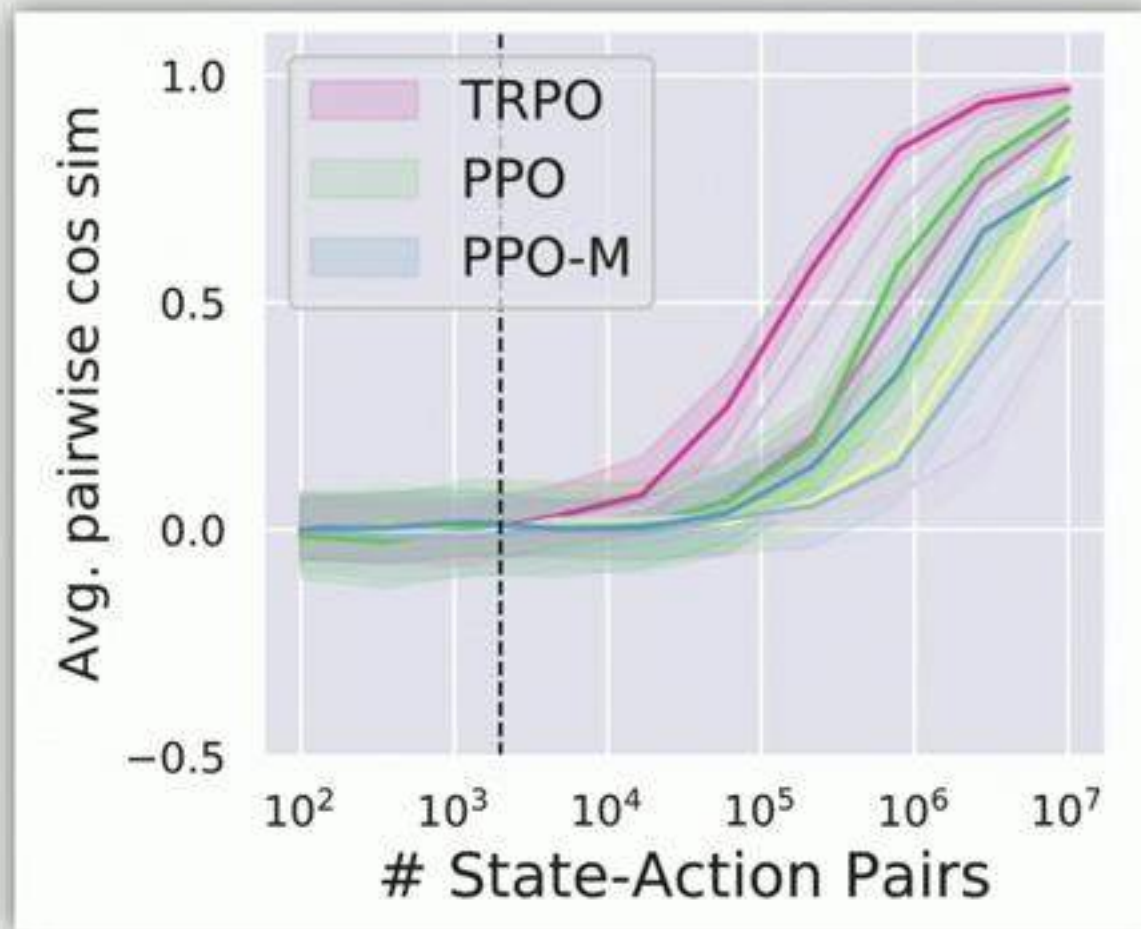
Gradient Estimation



Gradient Variance



Gradient Variance

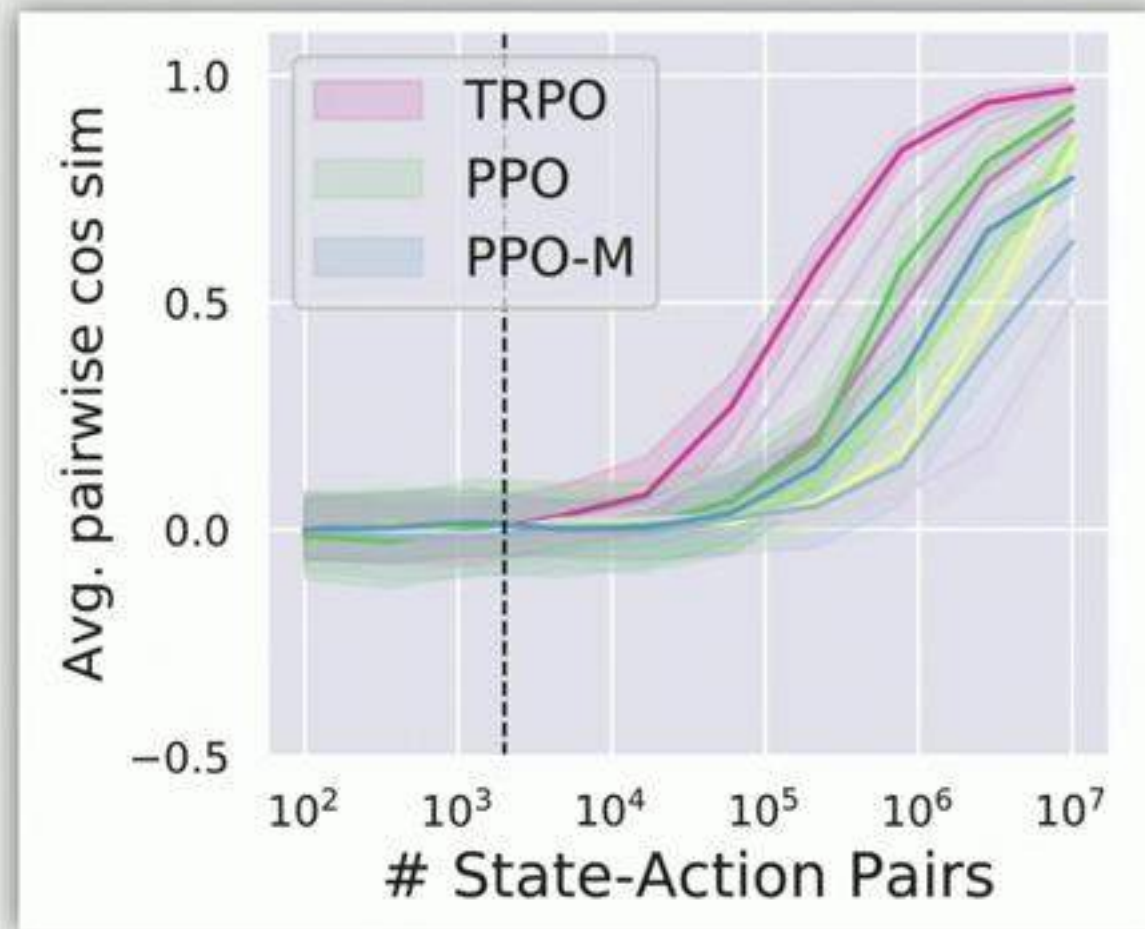


- ▶ **Black line:** relevant sample regime
- ▶ Gradients are less concentrated than they could be
- ▶ Less correlated for “harder” tasks, later iterations

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

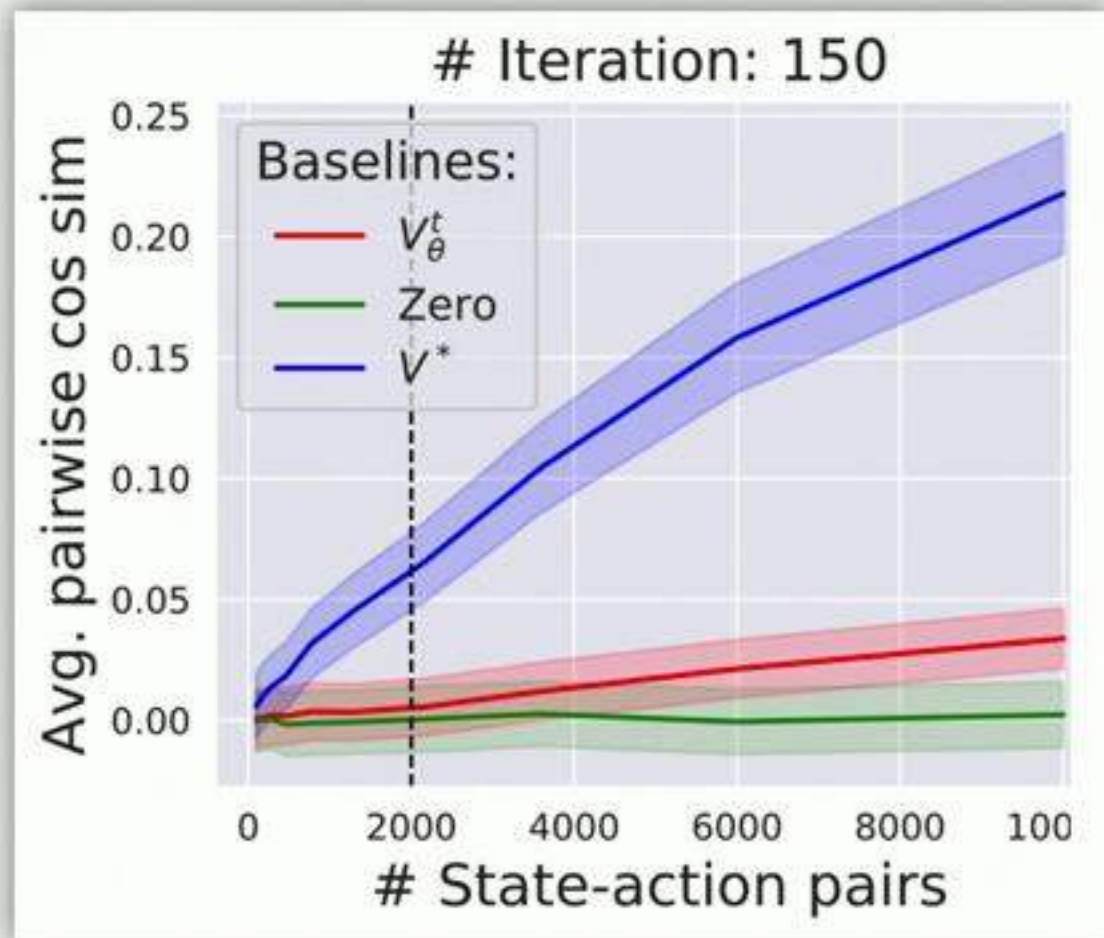
Value Prediction

Variance reduction needs **good value estimates**

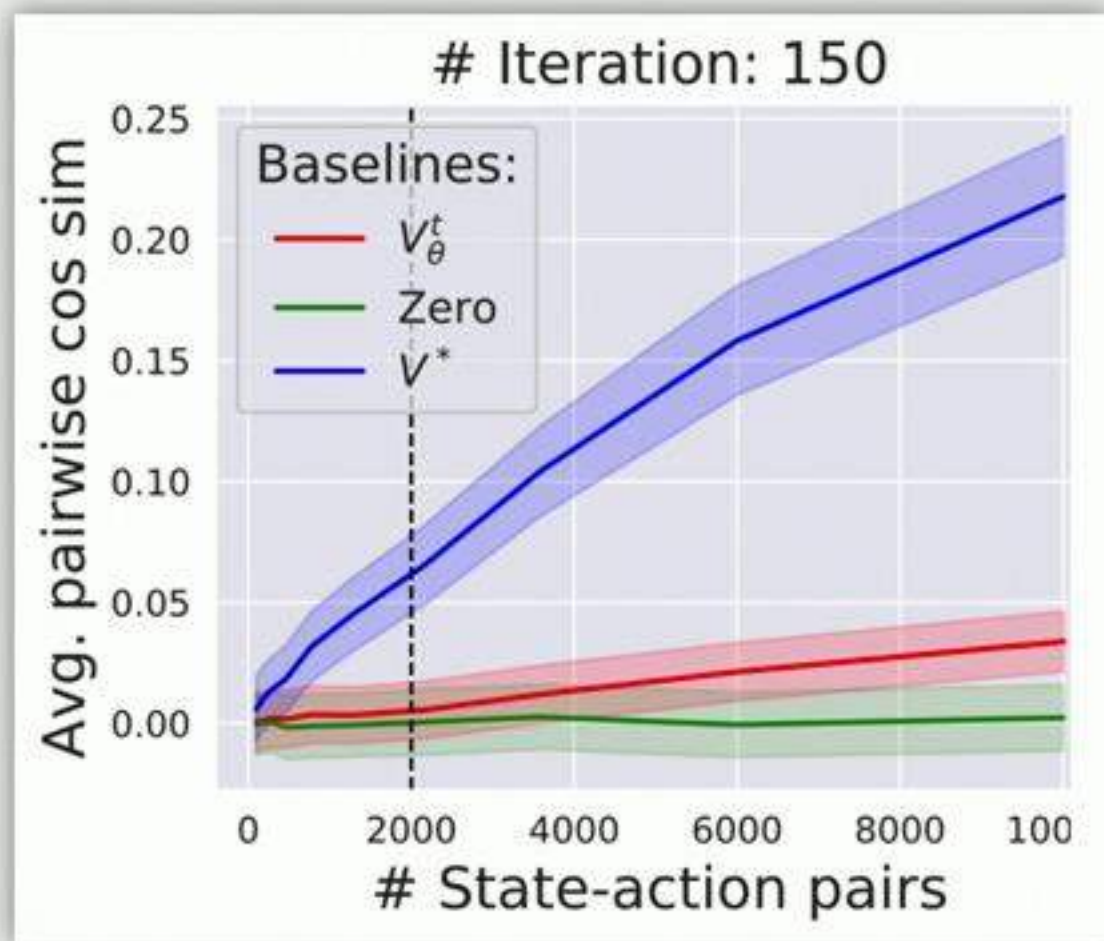
In Deep RL, values come from a neural network

To what degree do we actually reduce variance?

Value Prediction



Value Prediction



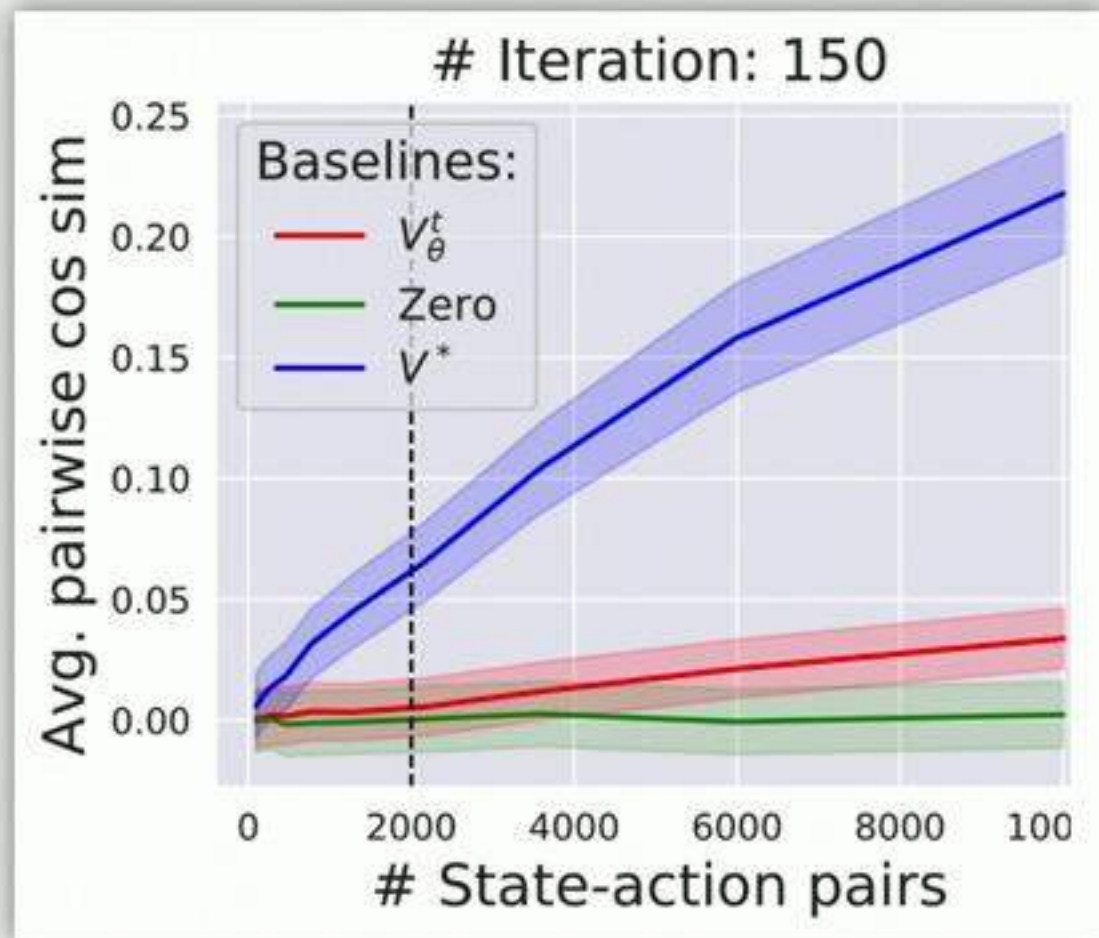
True value function

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?

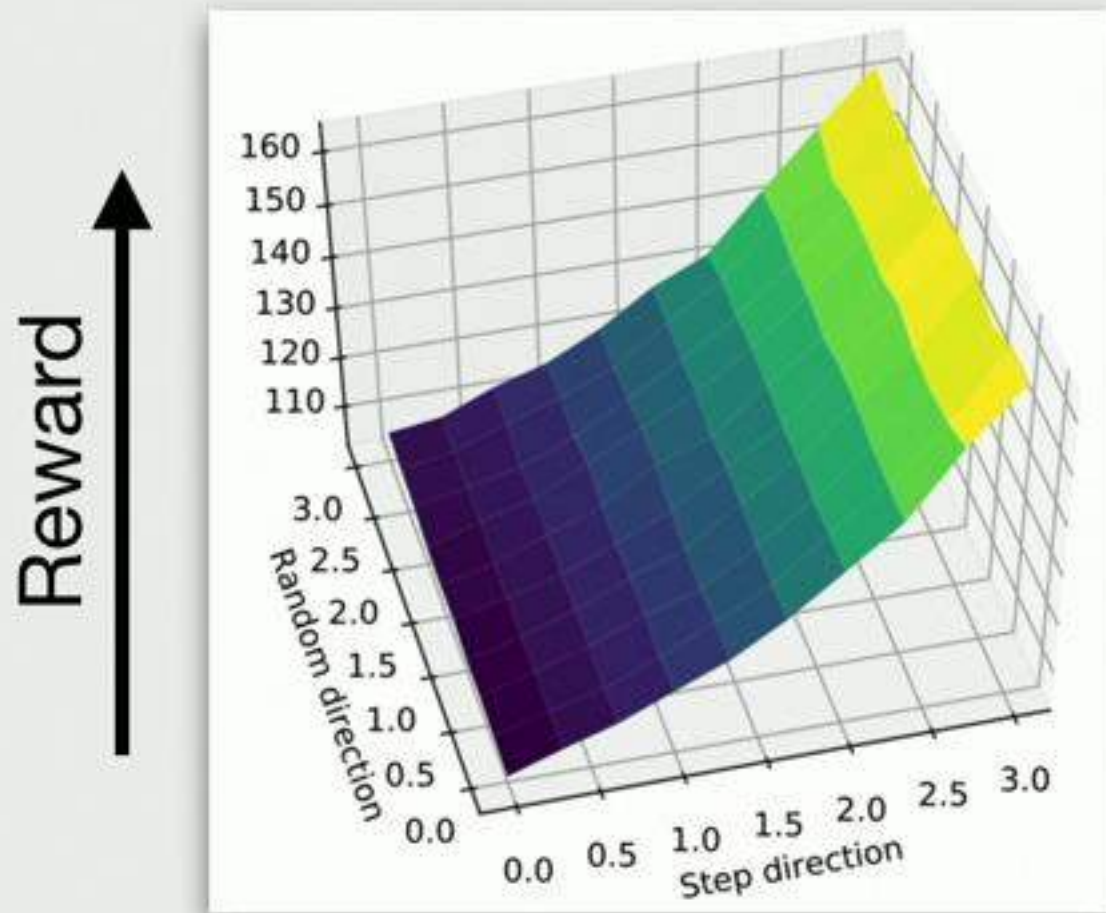
Optimization Landscapes

Assumption: taking gradient steps increases reward

How valid is this assumption in practice?

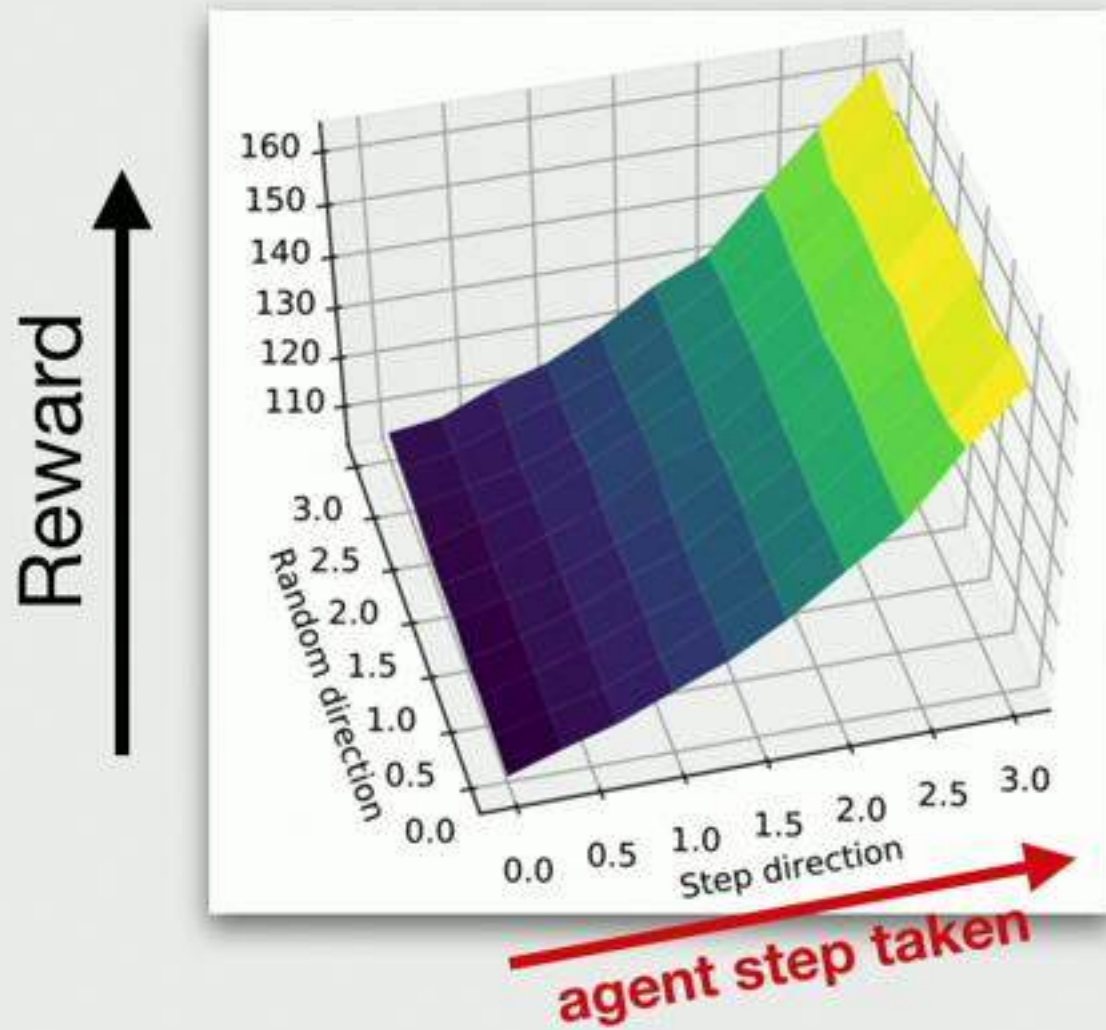
Optimization Landscapes

Step 0



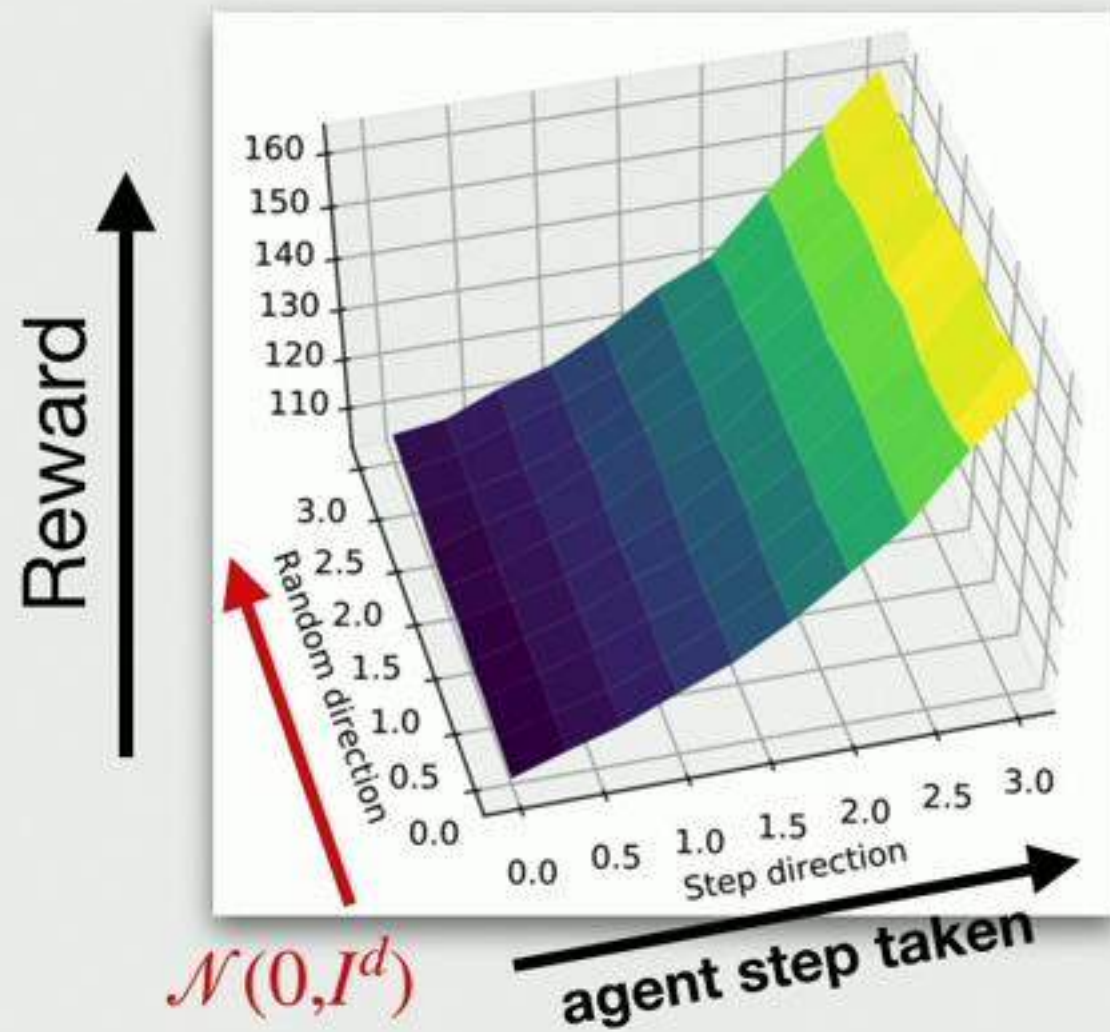
Optimization Landscapes

Step 0



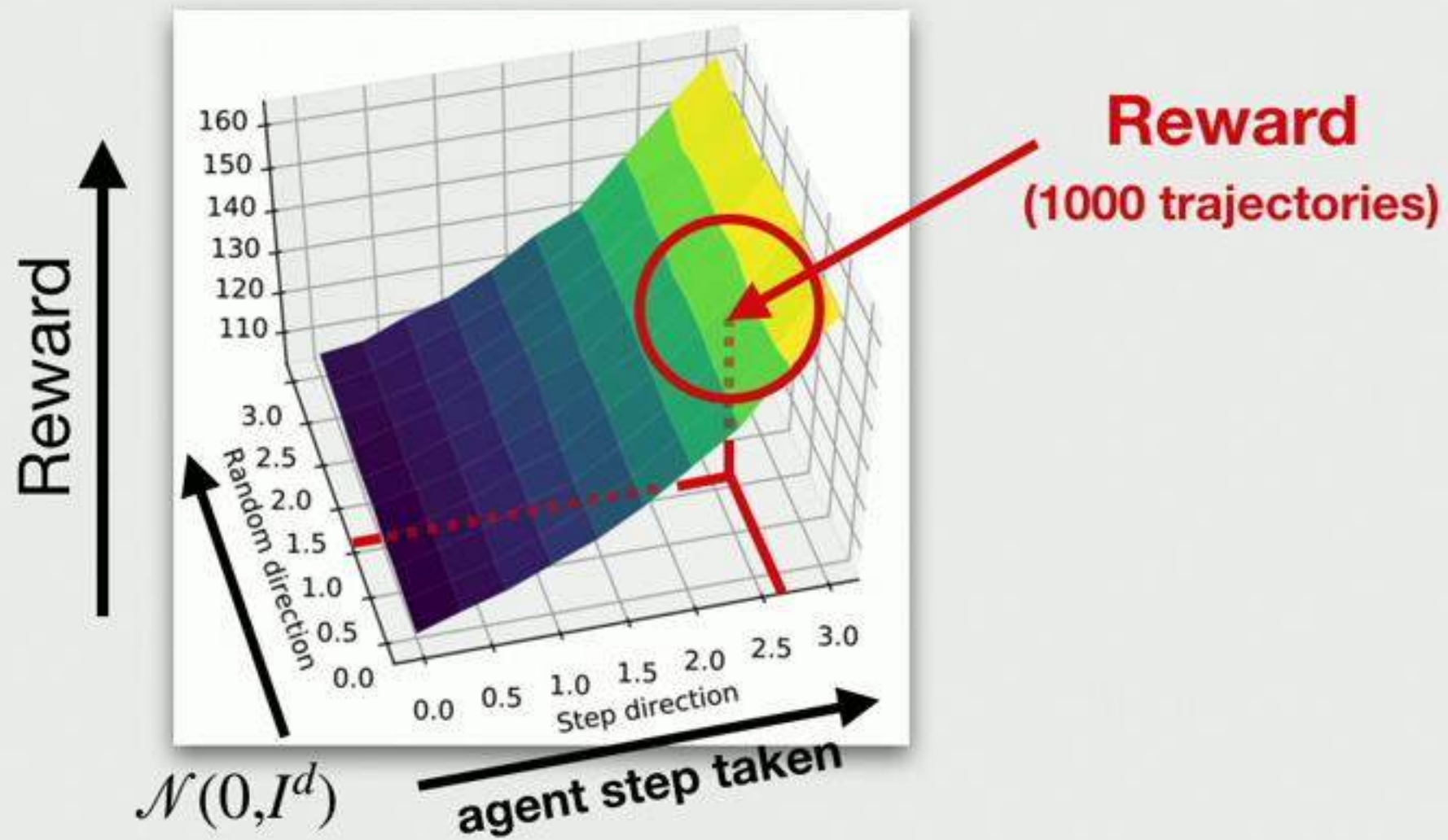
Optimization Landscapes

Step 0



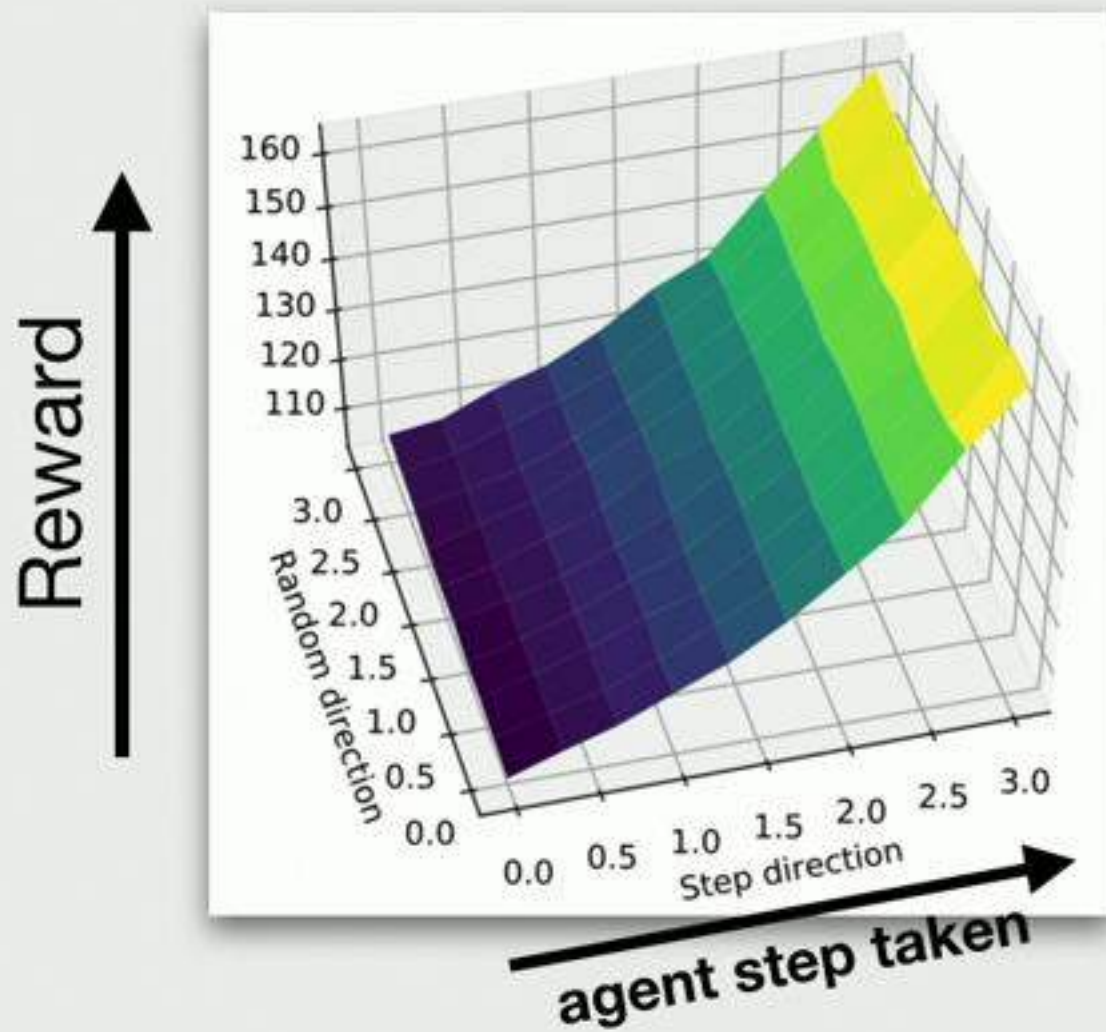
Optimization Landscapes

Step 0

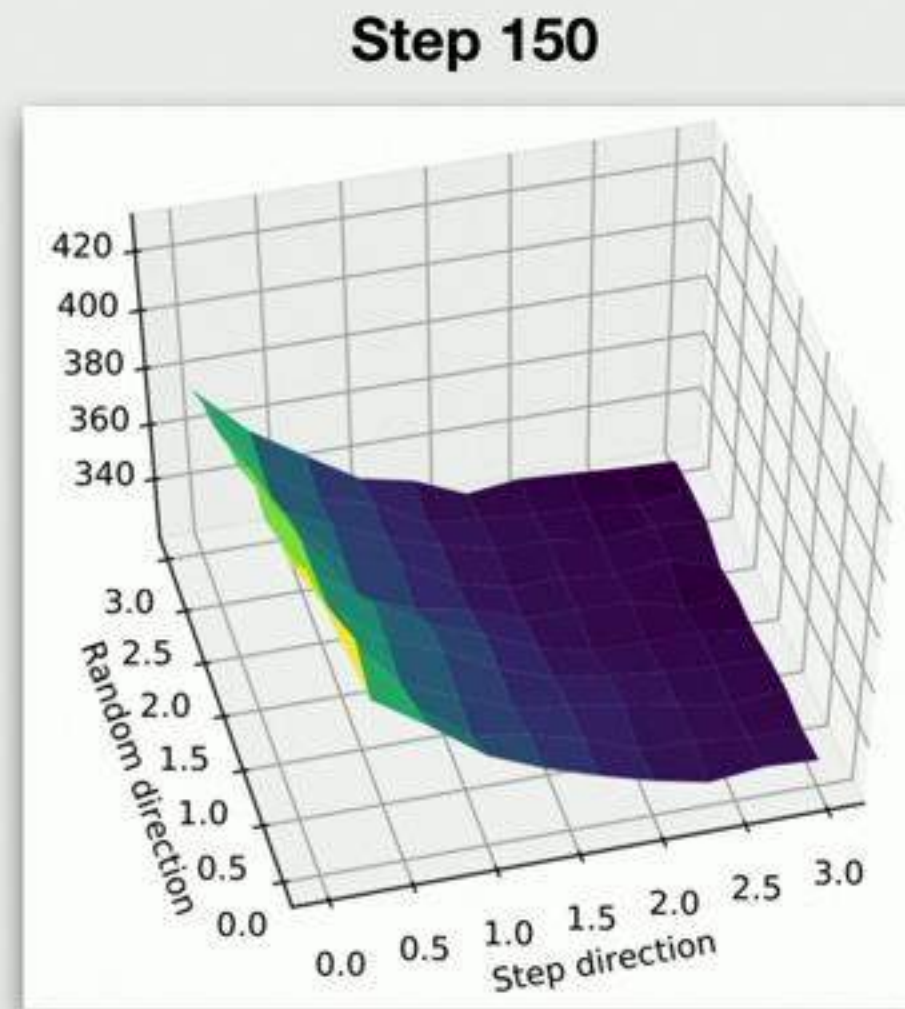
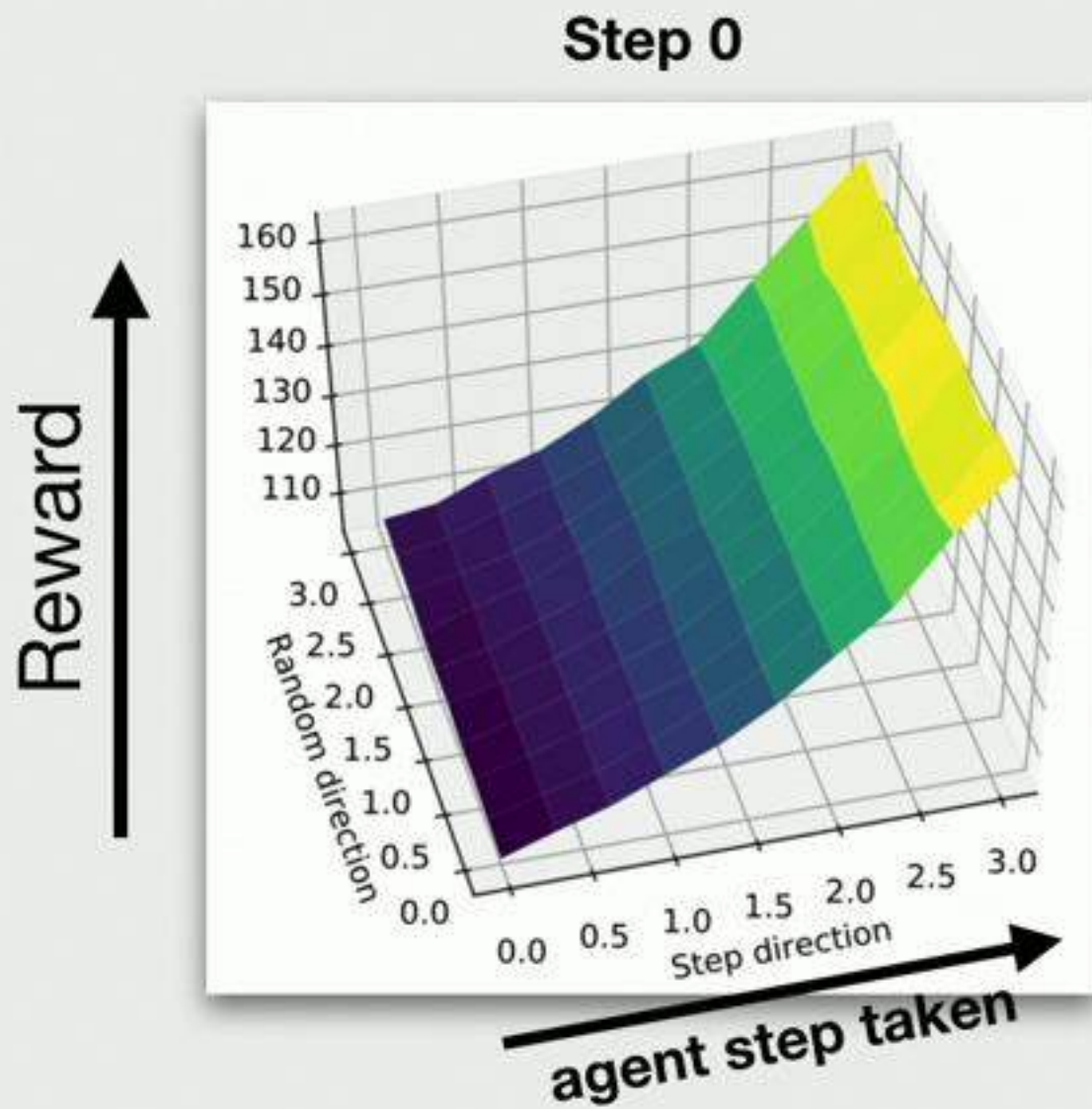


Optimization Landscapes

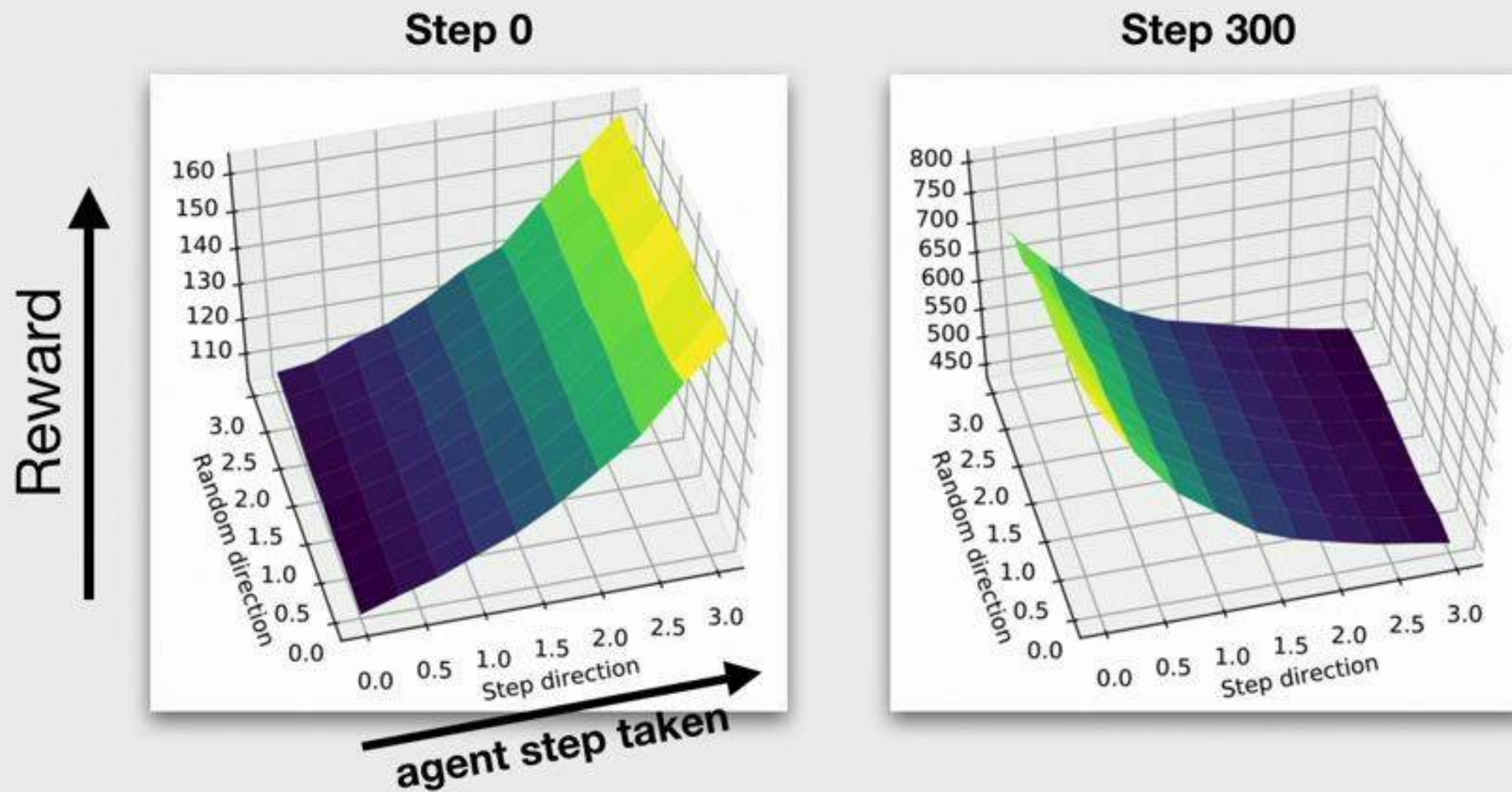
Step 0



Optimization Landscapes

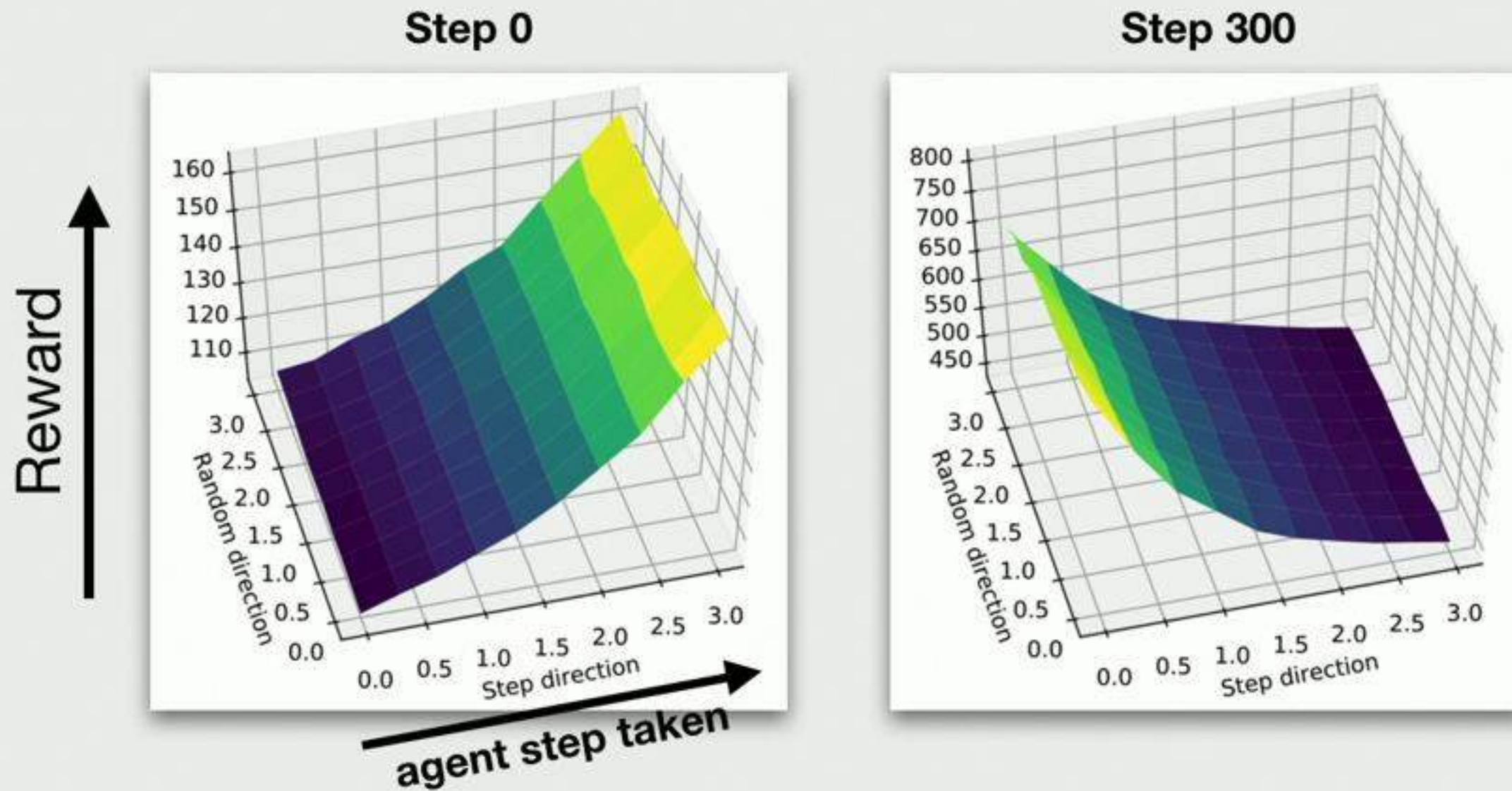


Optimization Landscapes



Steps are often not predictive

Optimization Landscapes



Steps are often not predictive

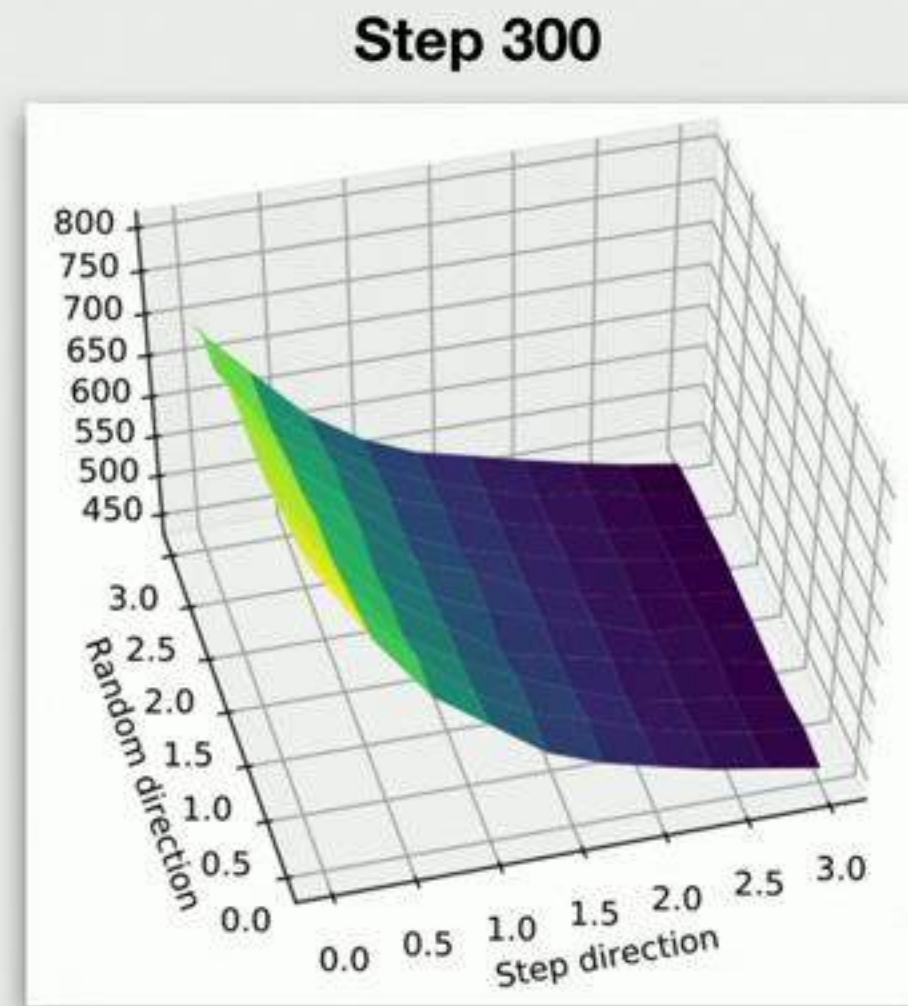
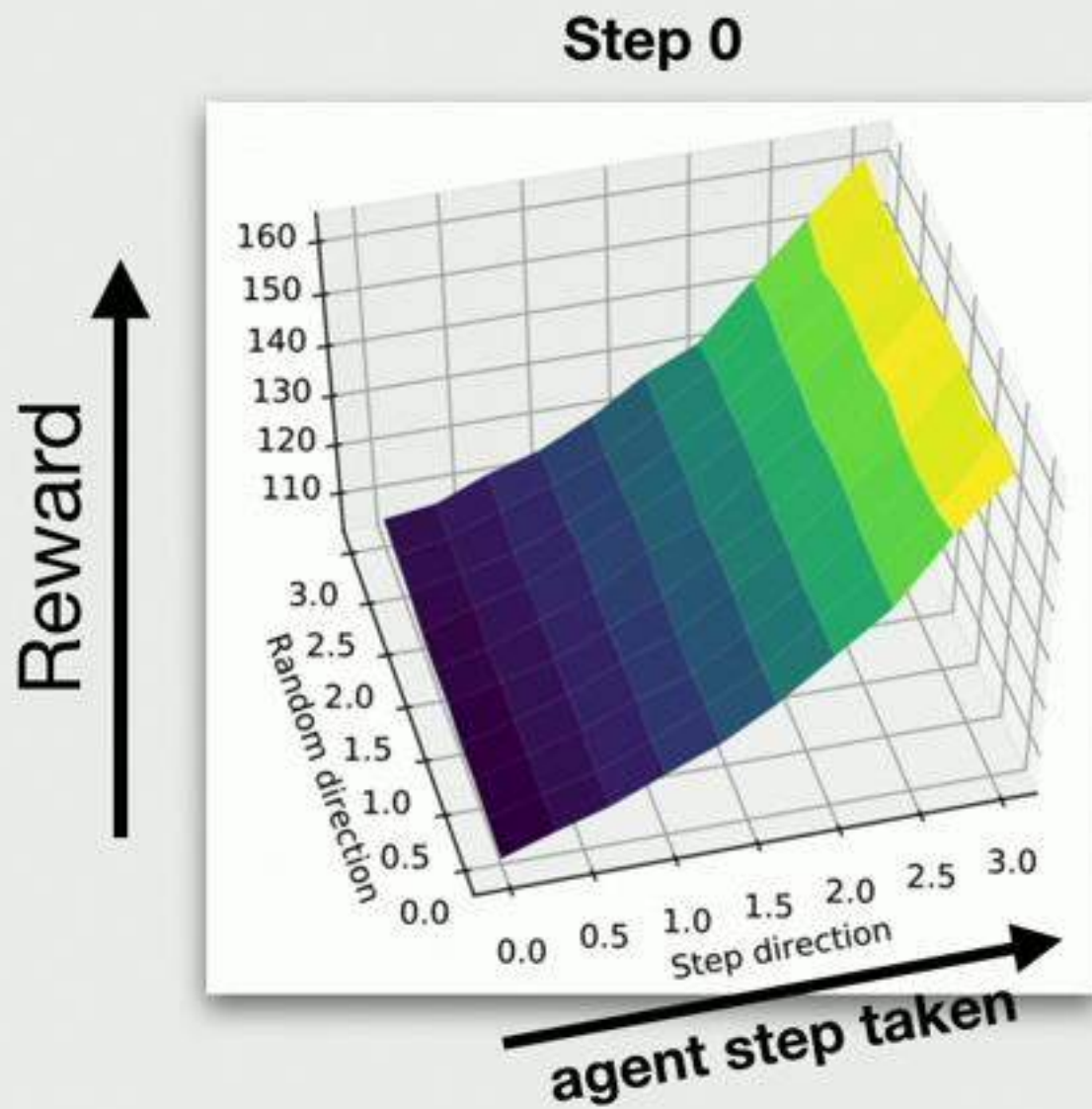
What's going on here?

Optimization Landscapes

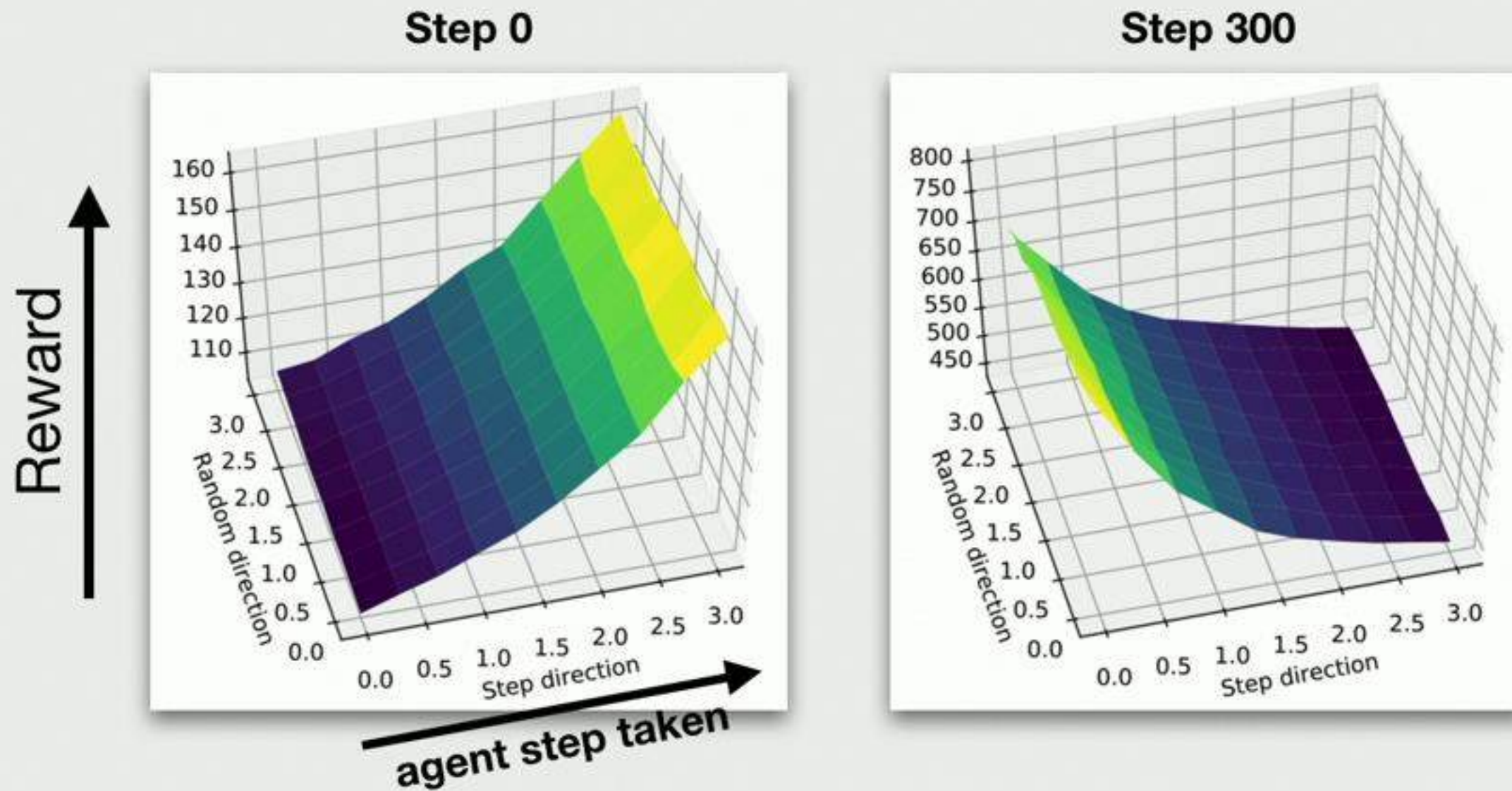
Methods iteratively maximize a “surrogate reward”

(not the true reward!)

Optimization Landscapes

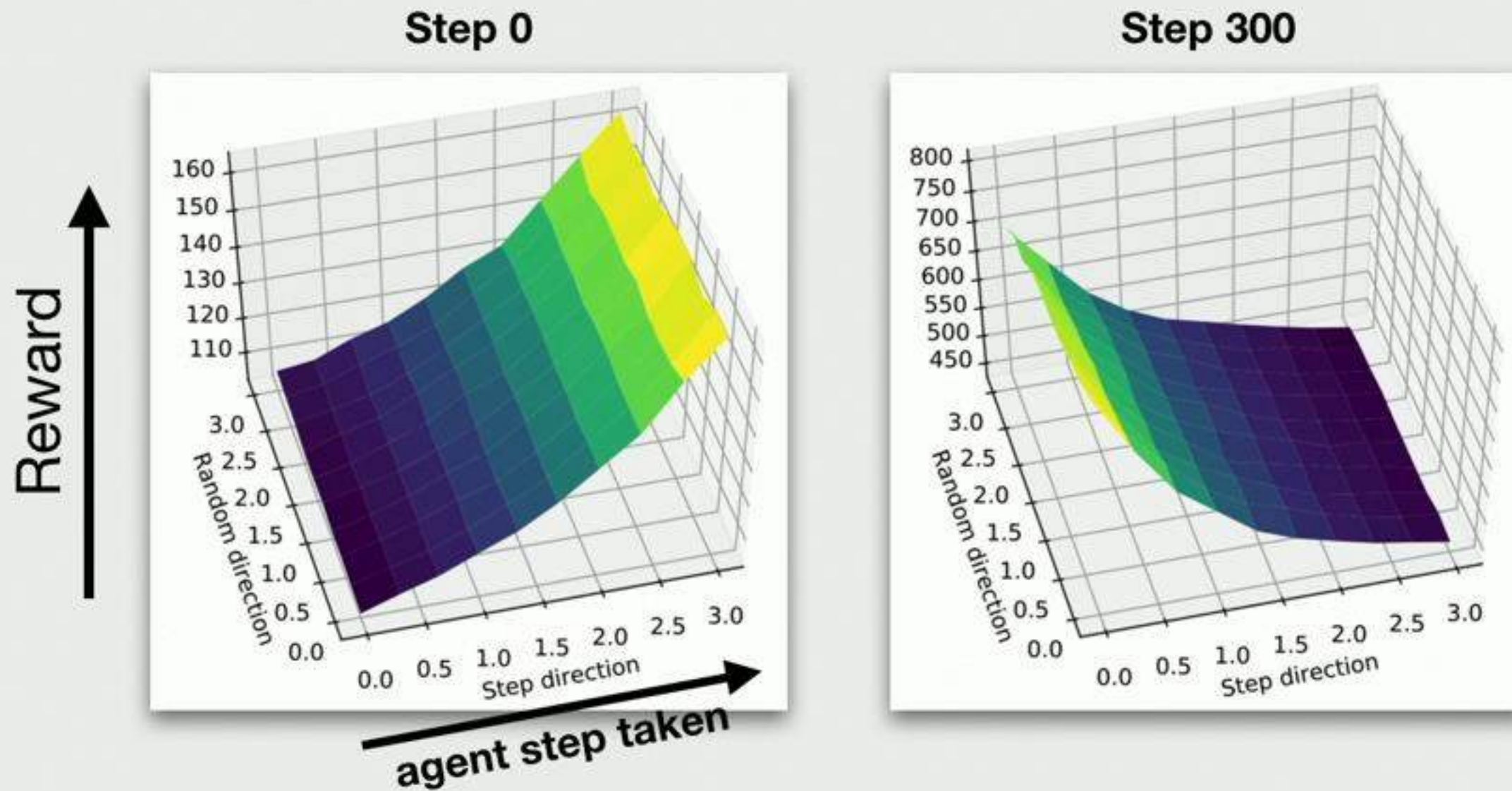


Optimization Landscapes



Steps are often not predictive

Optimization Landscapes



Steps are often not predictive

What's going on here?

Optimization Landscapes

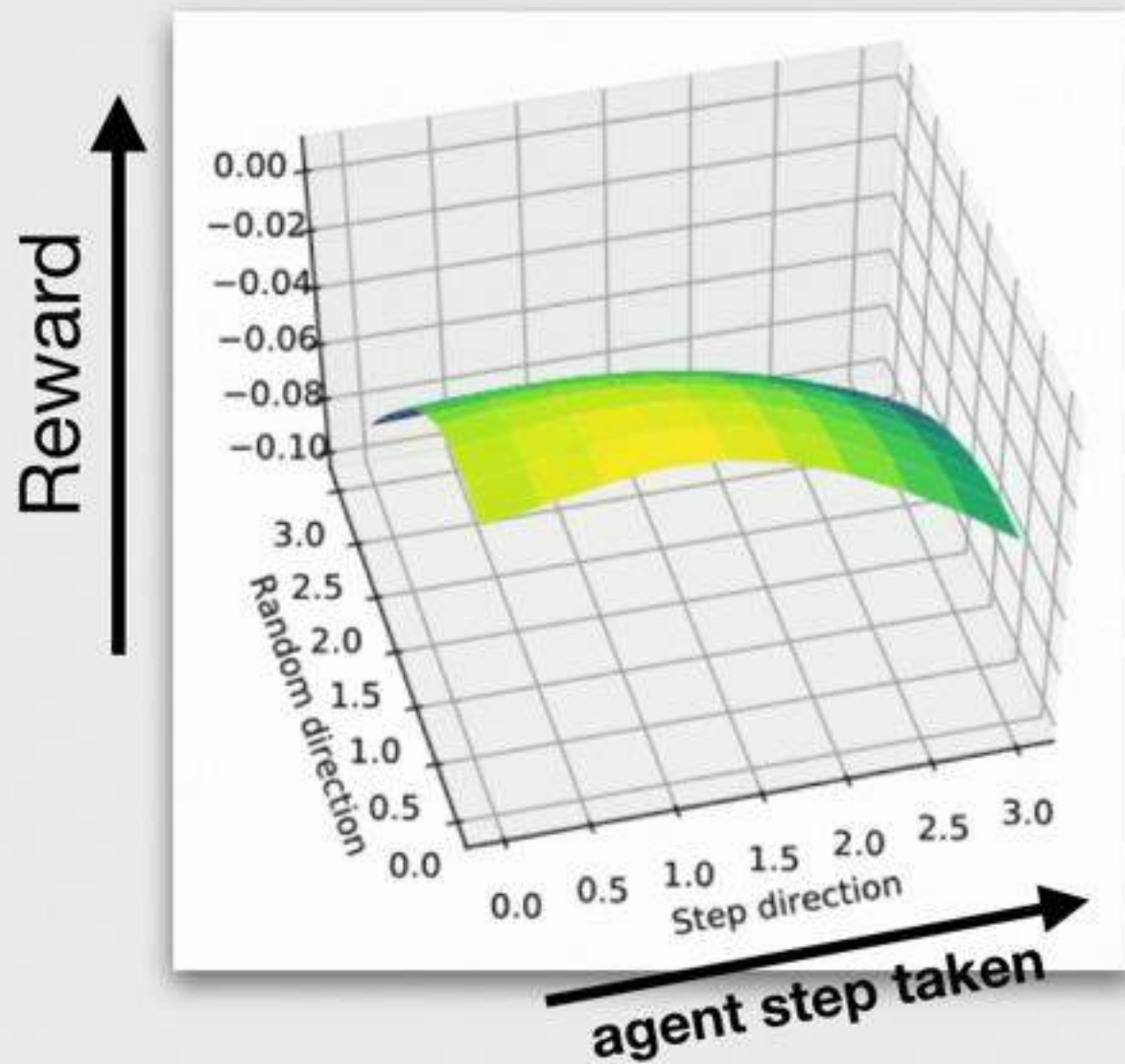
Methods iteratively maximize a “surrogate reward”

(not the true reward!)

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

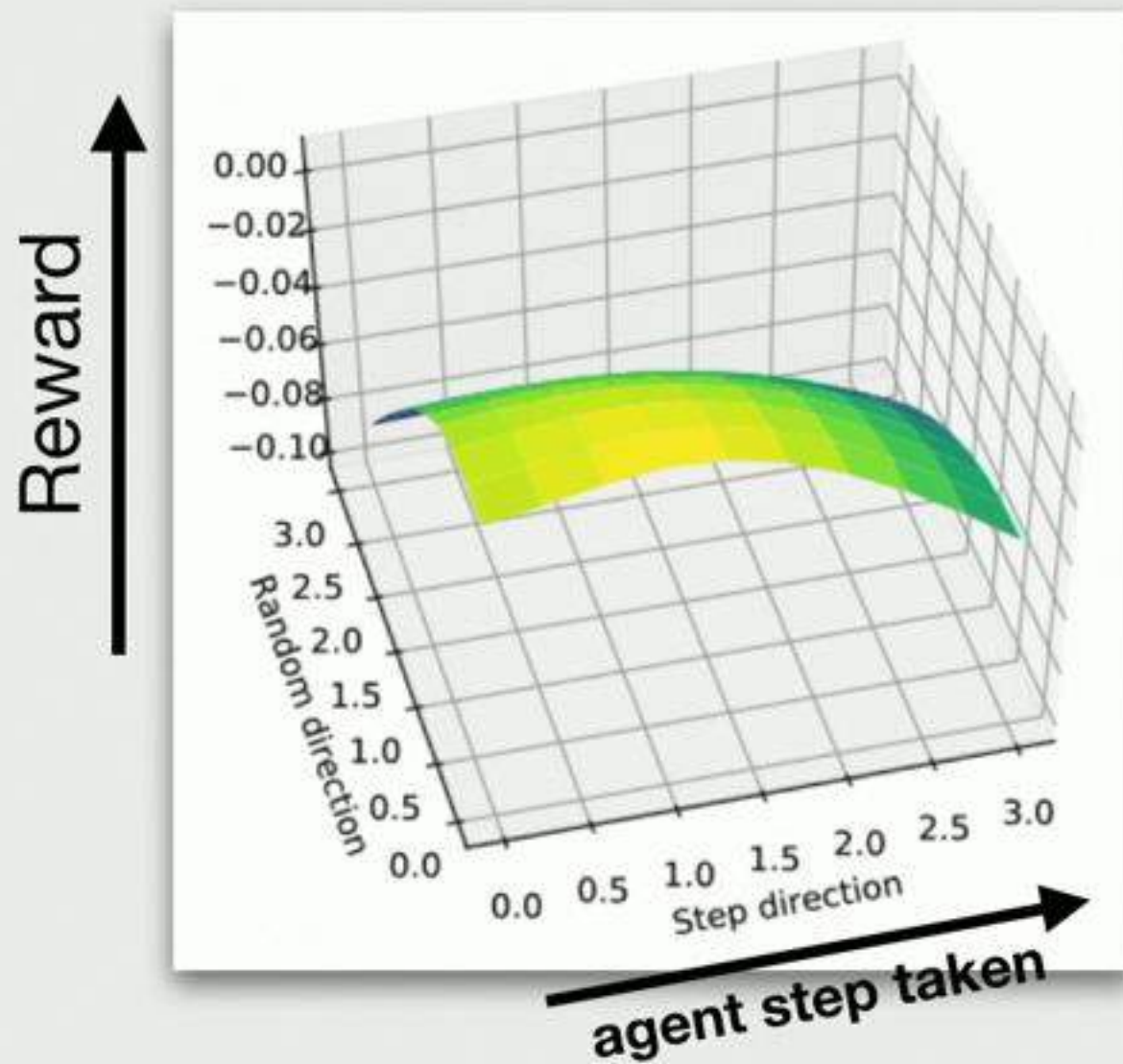
Optimization Landscapes

Surrogate Landscape

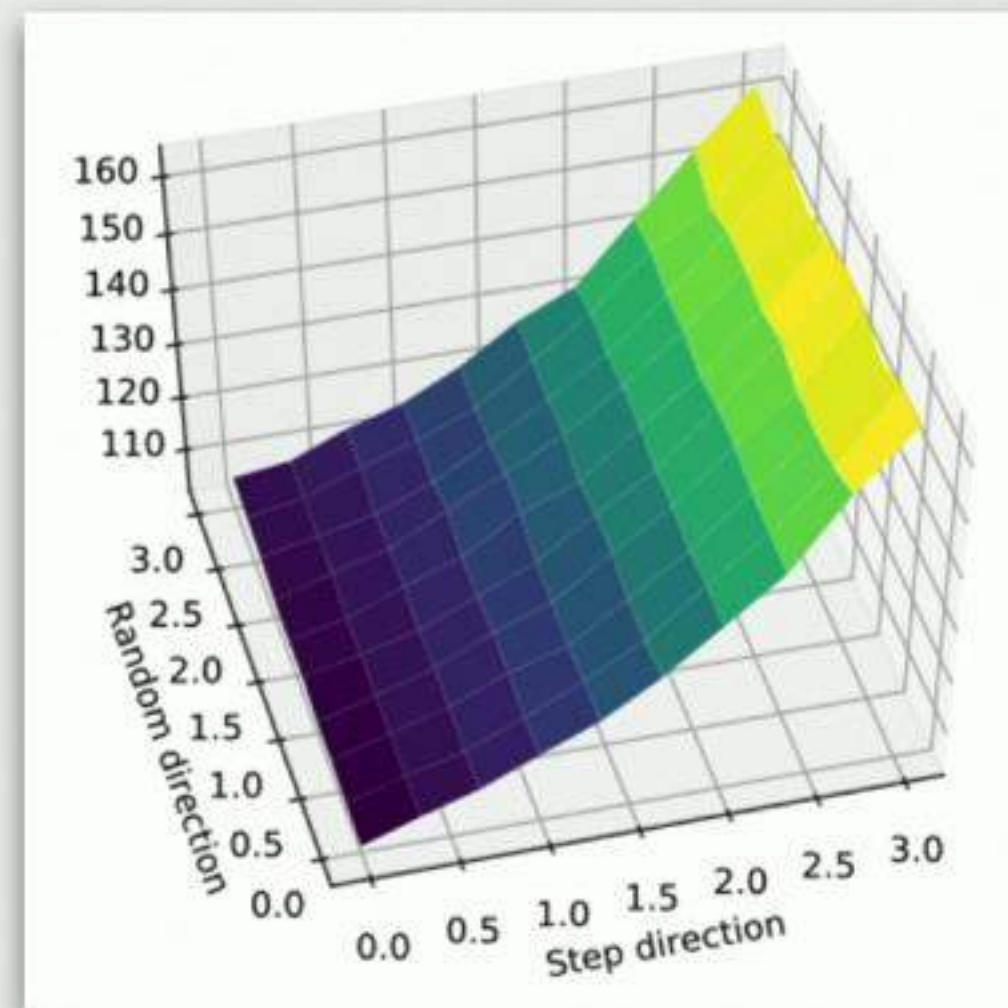


Optimization Landscapes

Surrogate Landscape



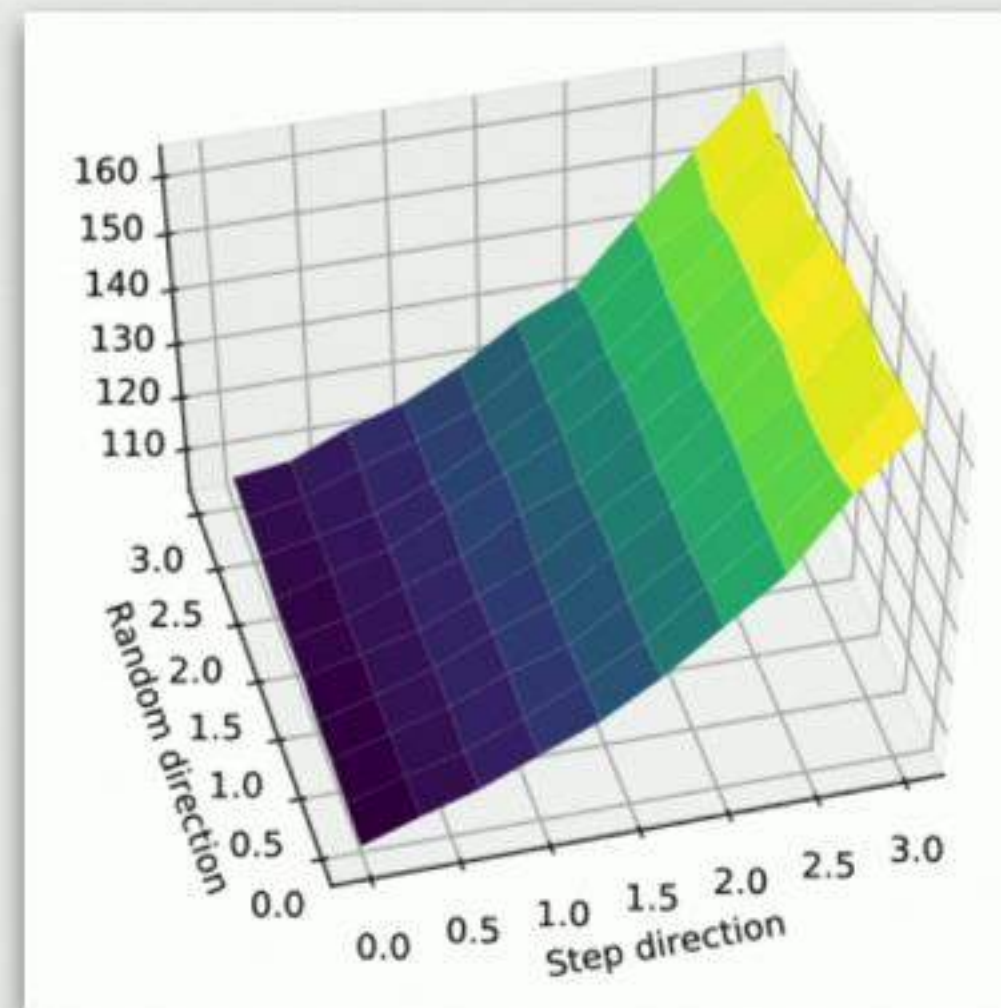
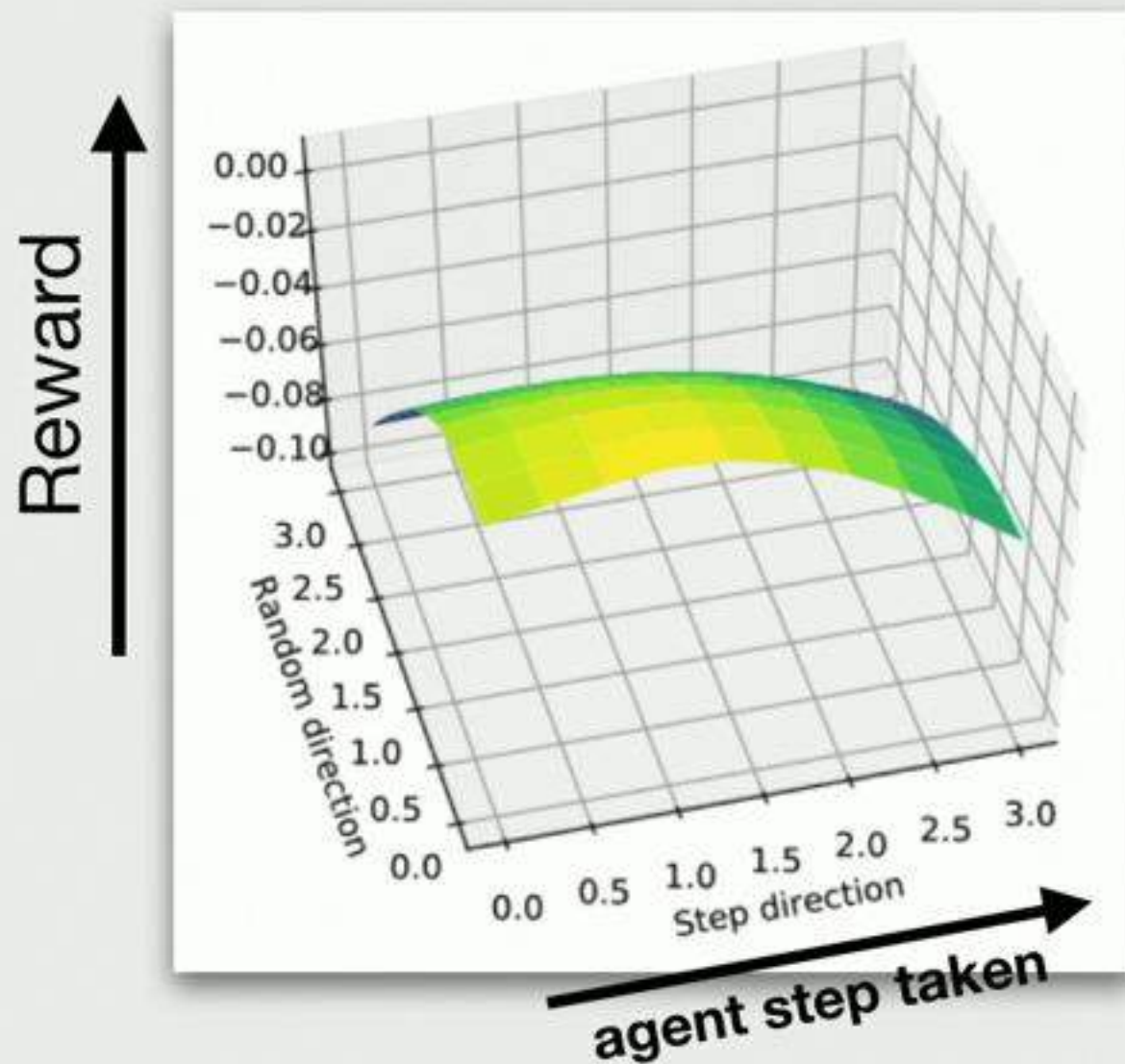
Reward Landscape



Optimization Landscapes

Surrogate Landscape

Reward Landscape

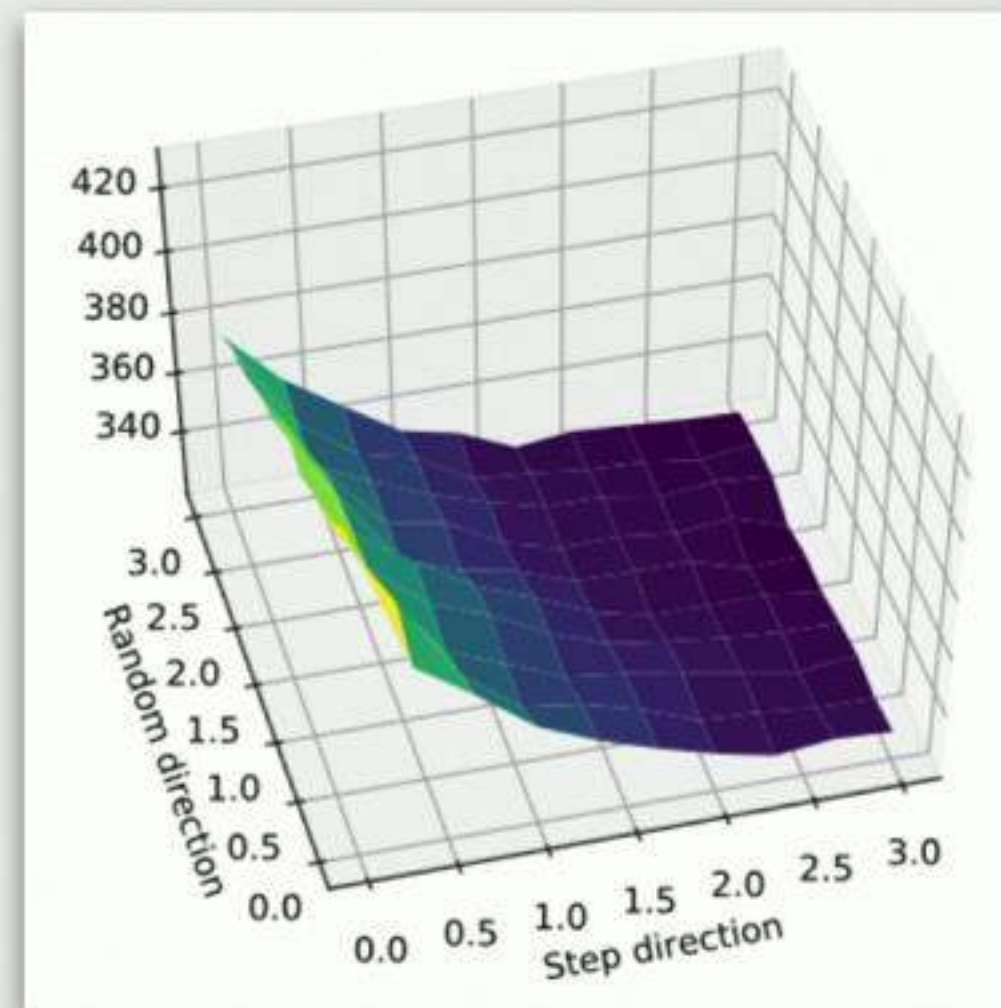
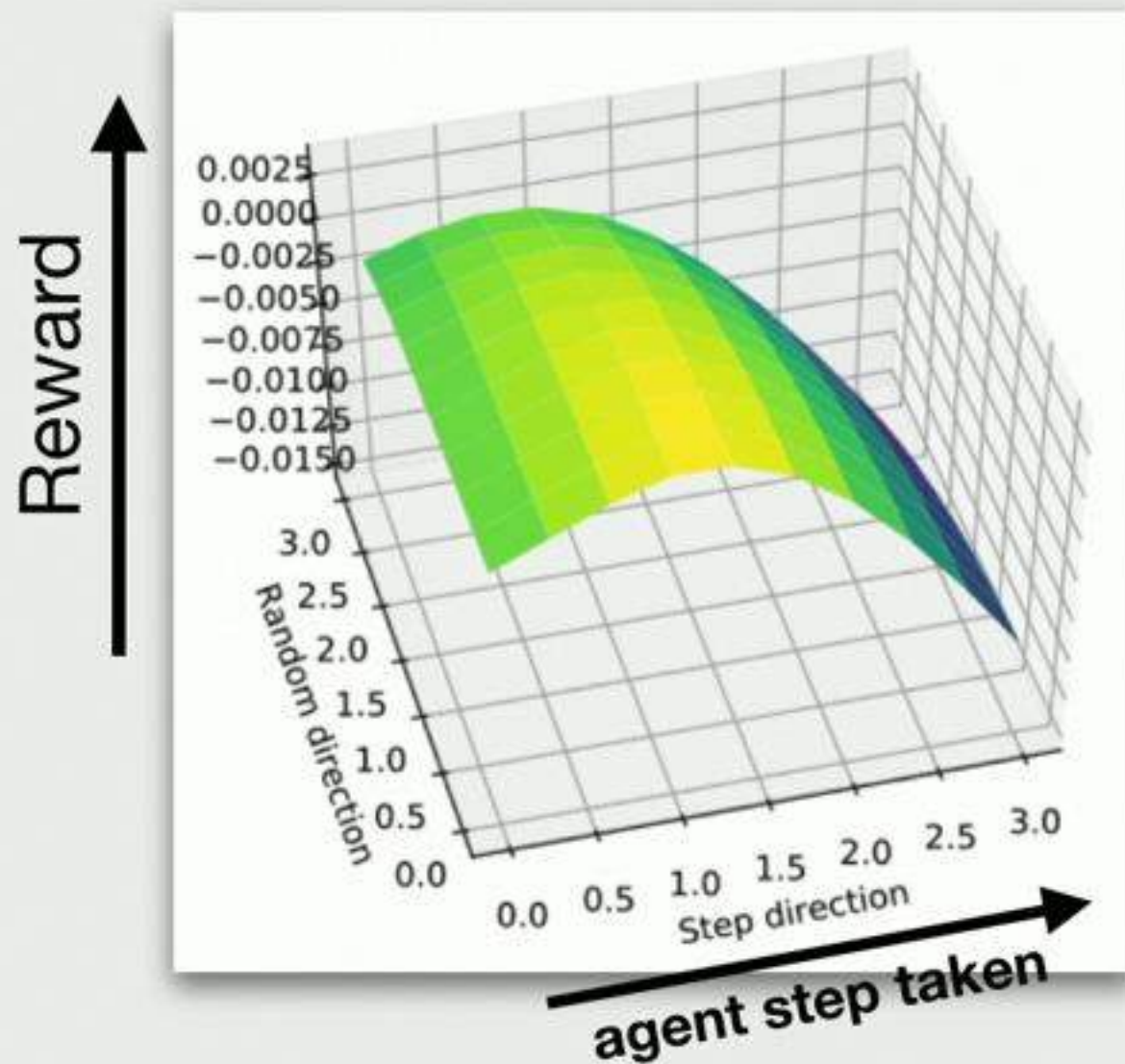


Step 0

Optimization Landscapes

Surrogate Landscape

Reward Landscape

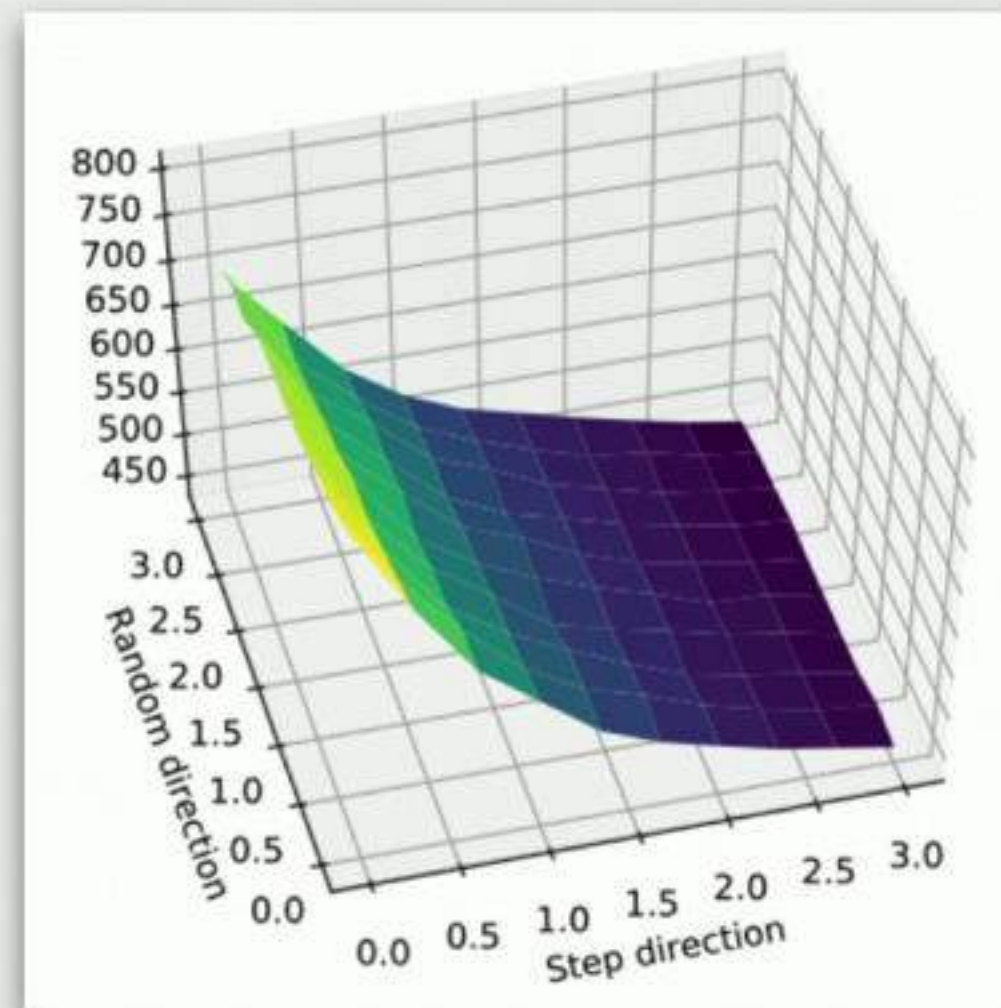
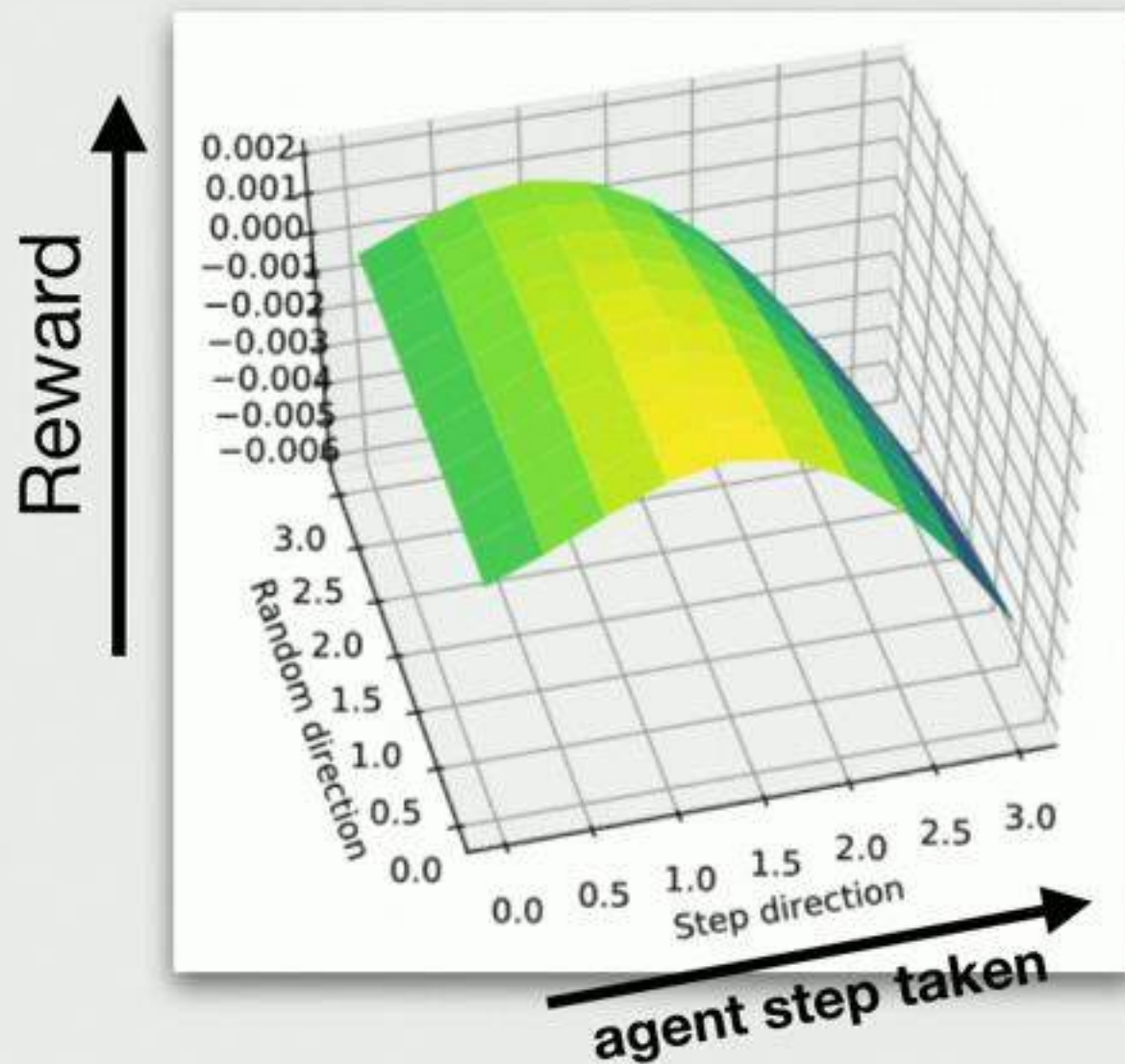


Step 150

Optimization Landscapes

Surrogate Landscape

Reward Landscape

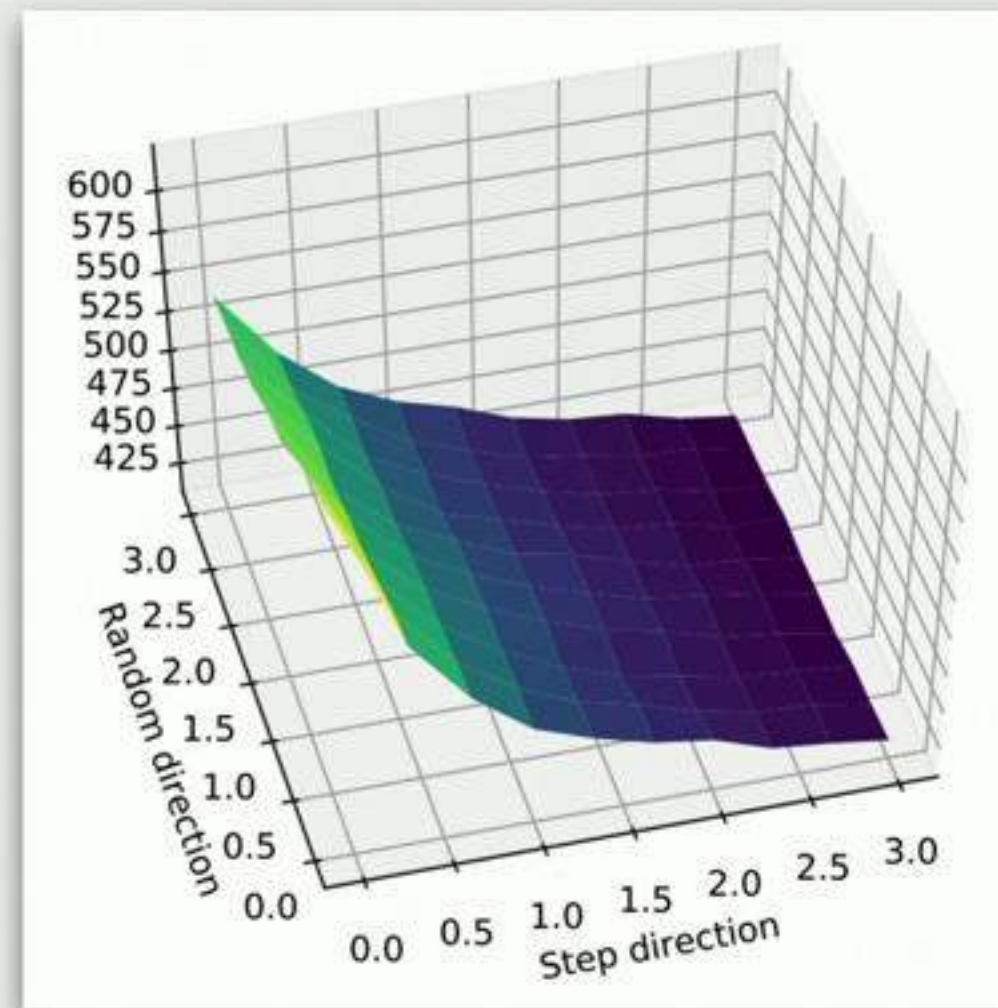
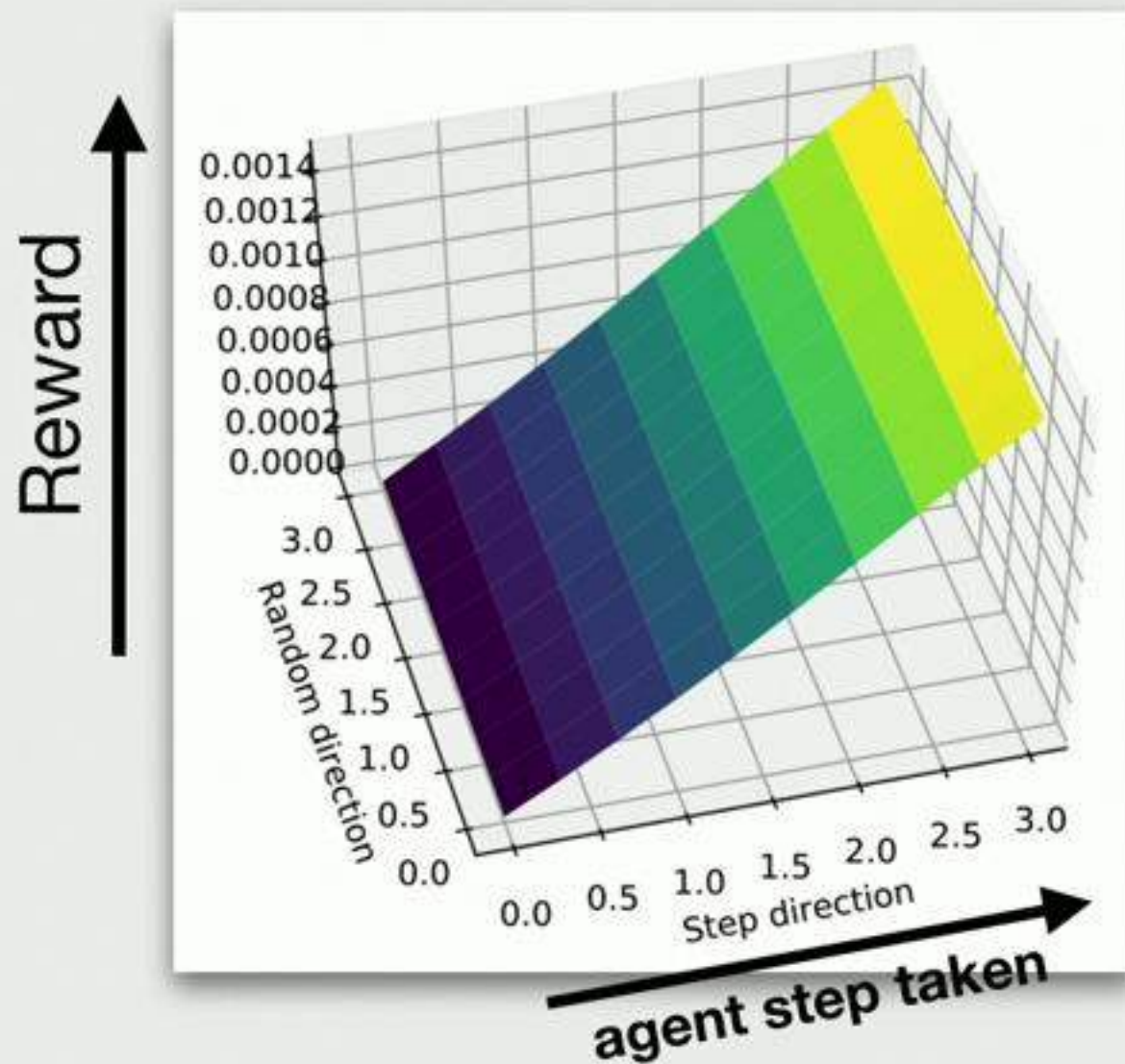


Step 300

Optimization Landscapes

Surrogate Landscape

Reward Landscape

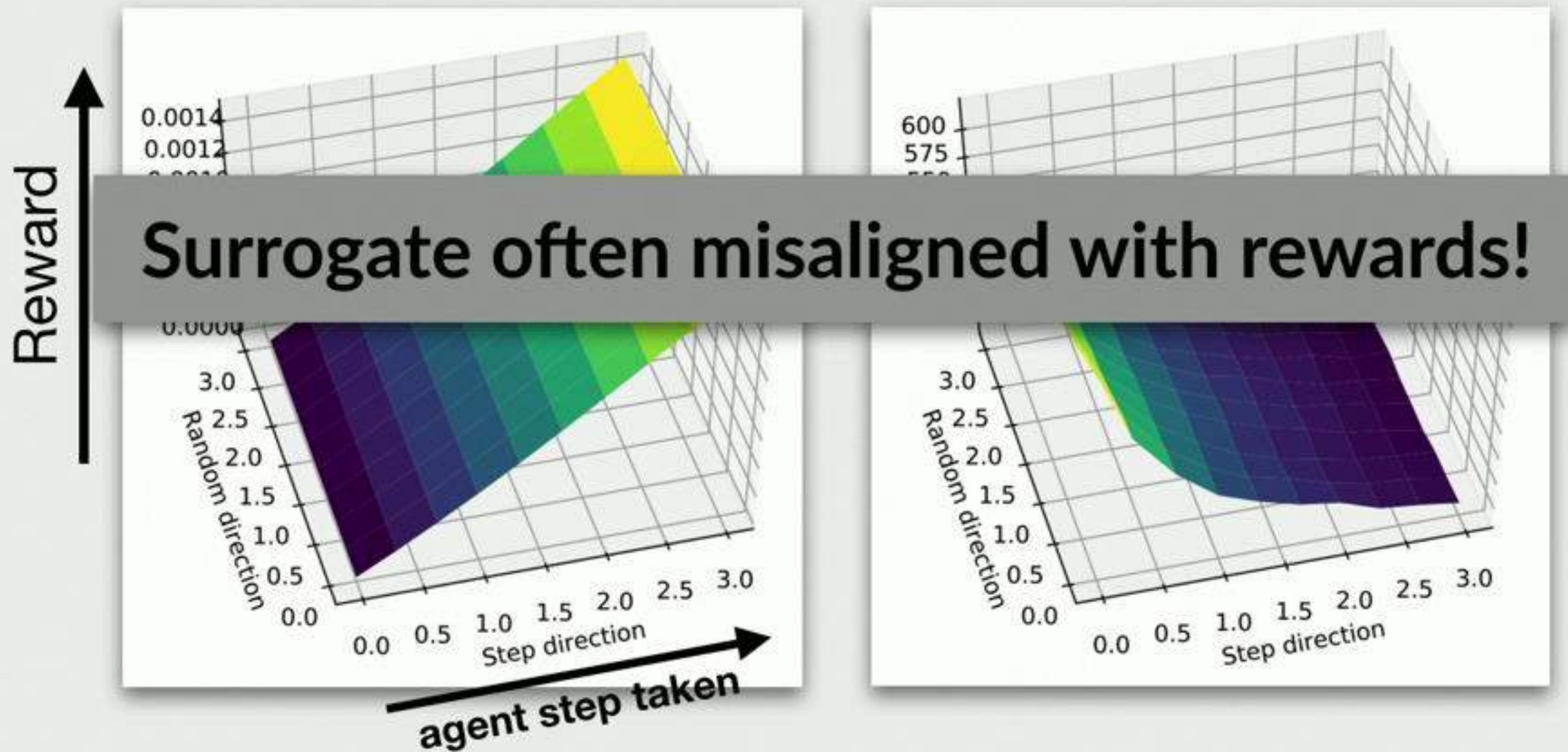


Step 450

Optimization Landscapes

Surrogate Landscape

Reward Landscape

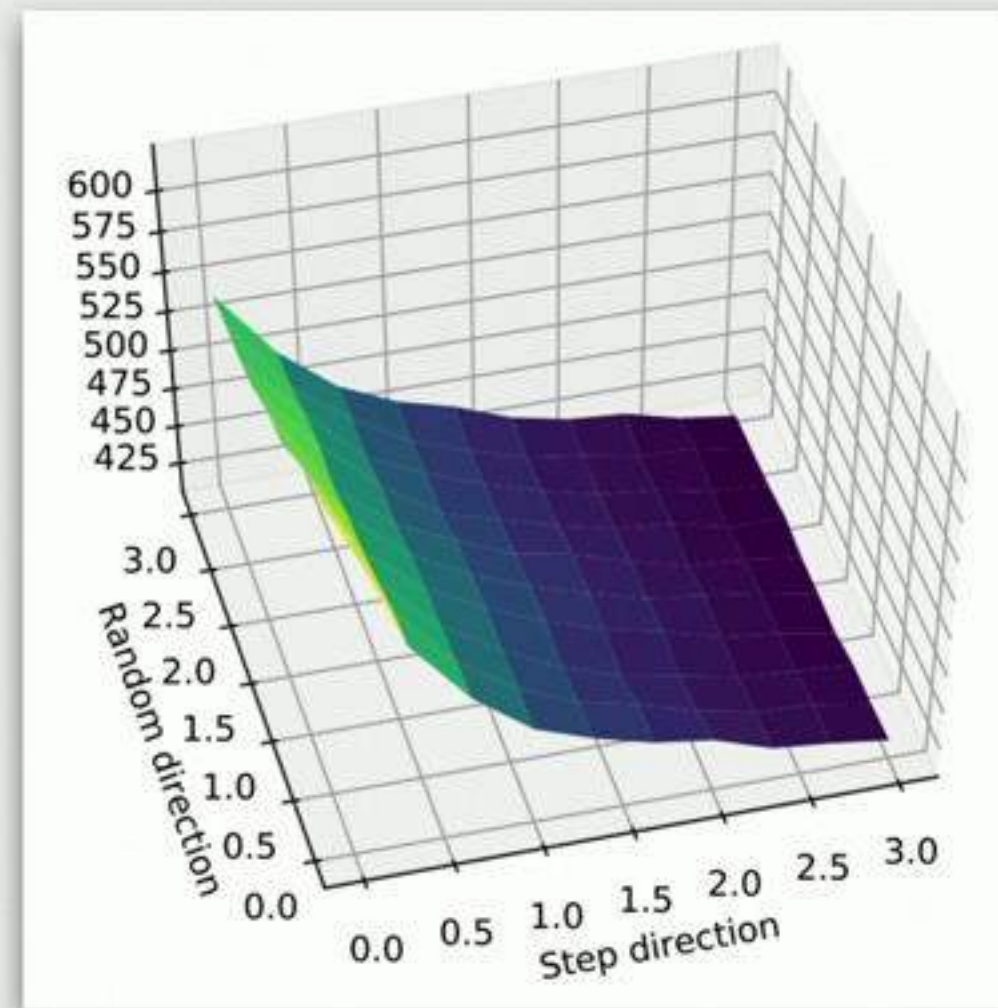
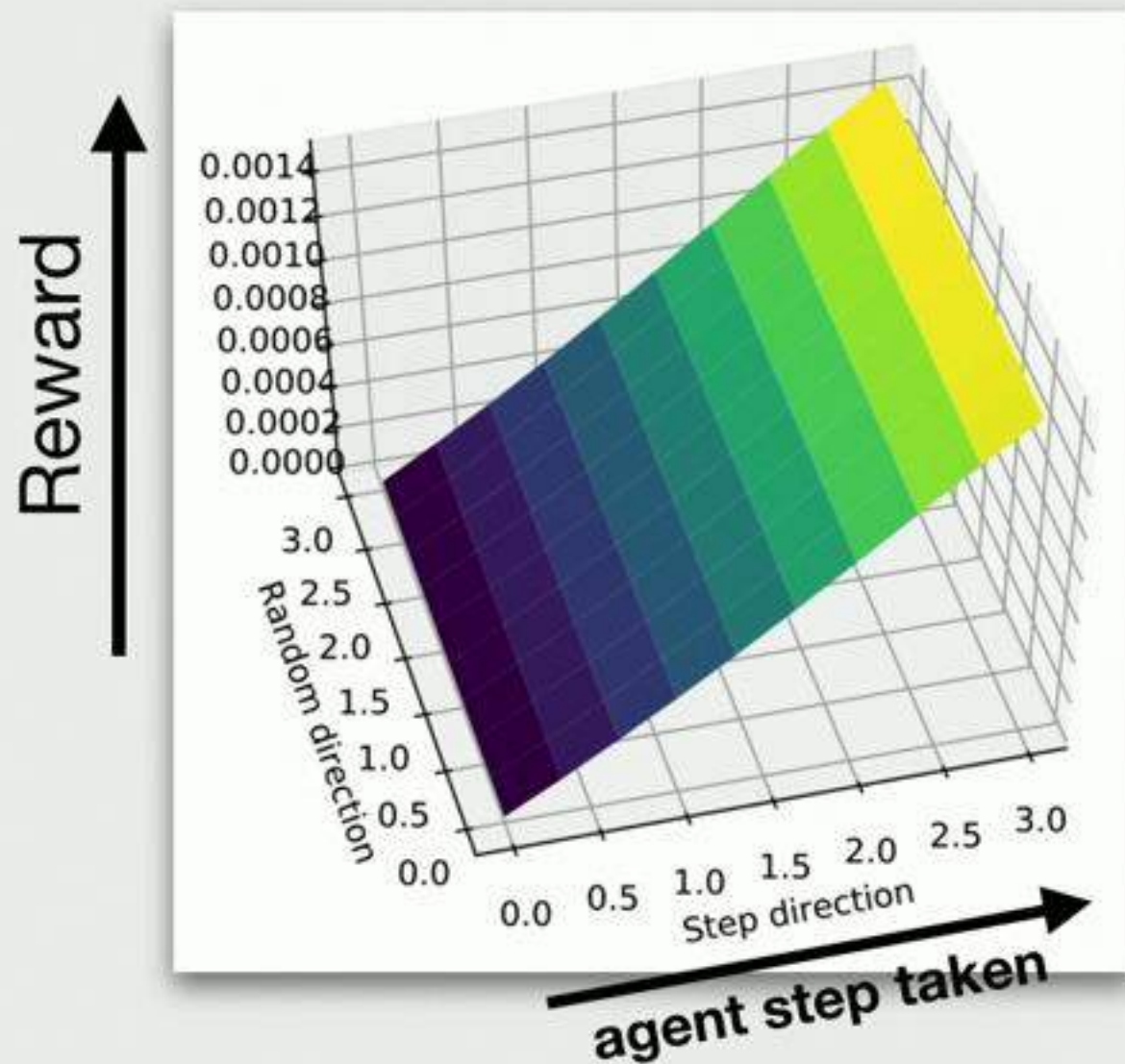


Step 450

Optimization Landscapes

Surrogate Landscape

Reward Landscape

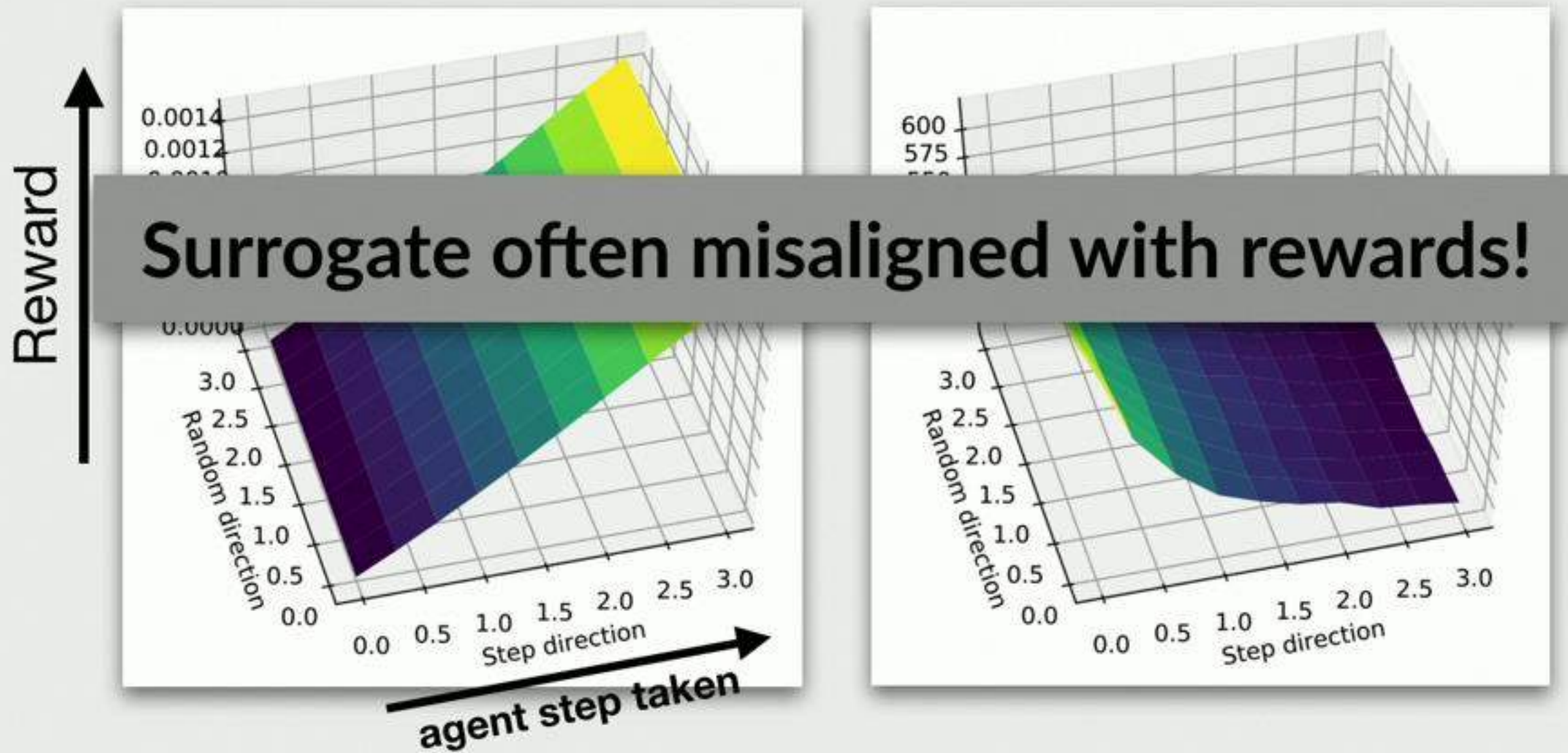


Step 450

Optimization Landscapes

Surrogate Landscape

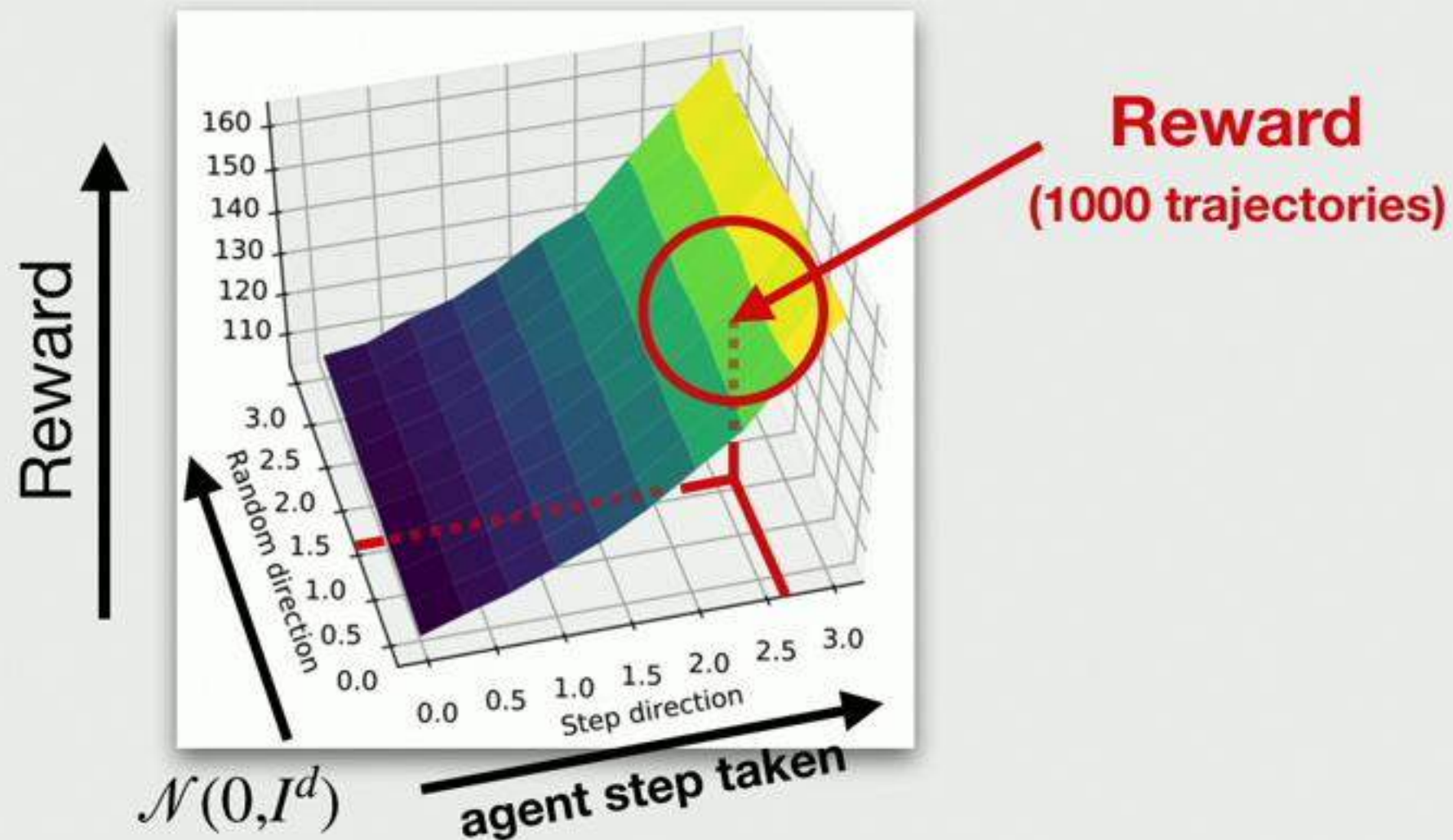
Reward Landscape



Step 450

Optimization Landscapes

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

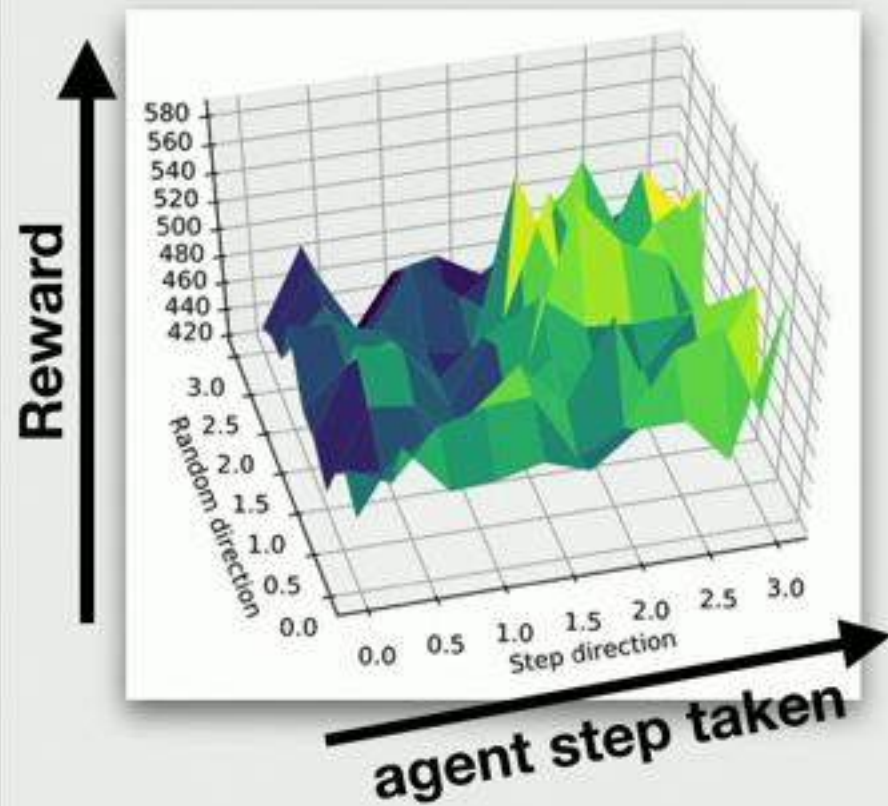


How do landscapes appear to the agent?

(~20 trajectories)

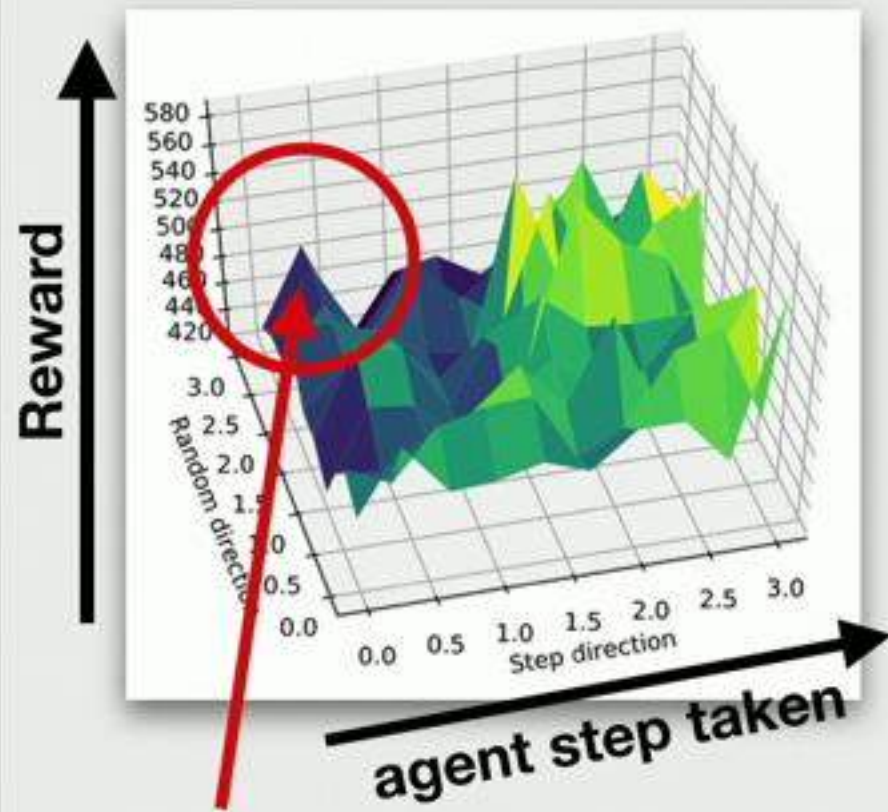
Optimization Landscapes

20-sample estimates



Optimization Landscapes

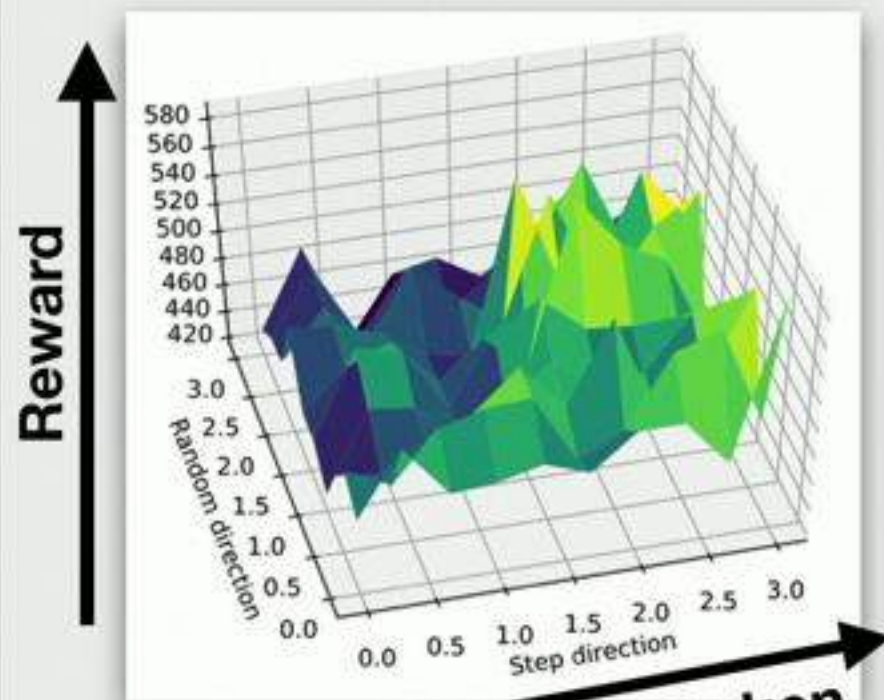
20-sample estimates



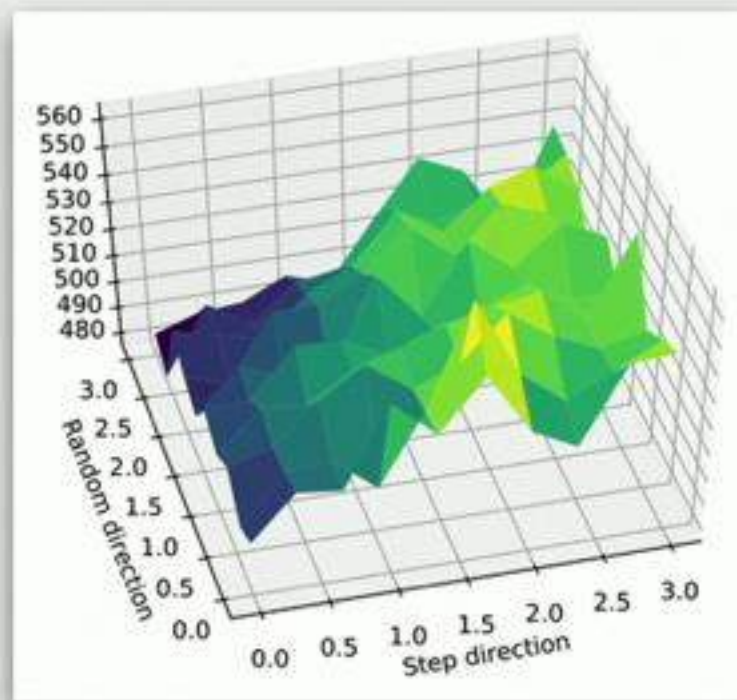
20 trajectories per reward estimate

Optimization Landscapes

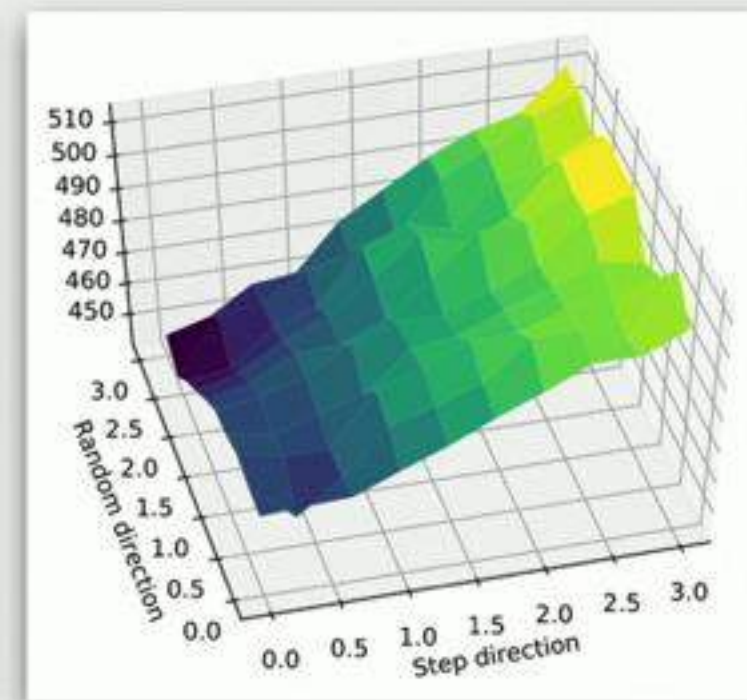
20-sample estimates



200-sample estimates



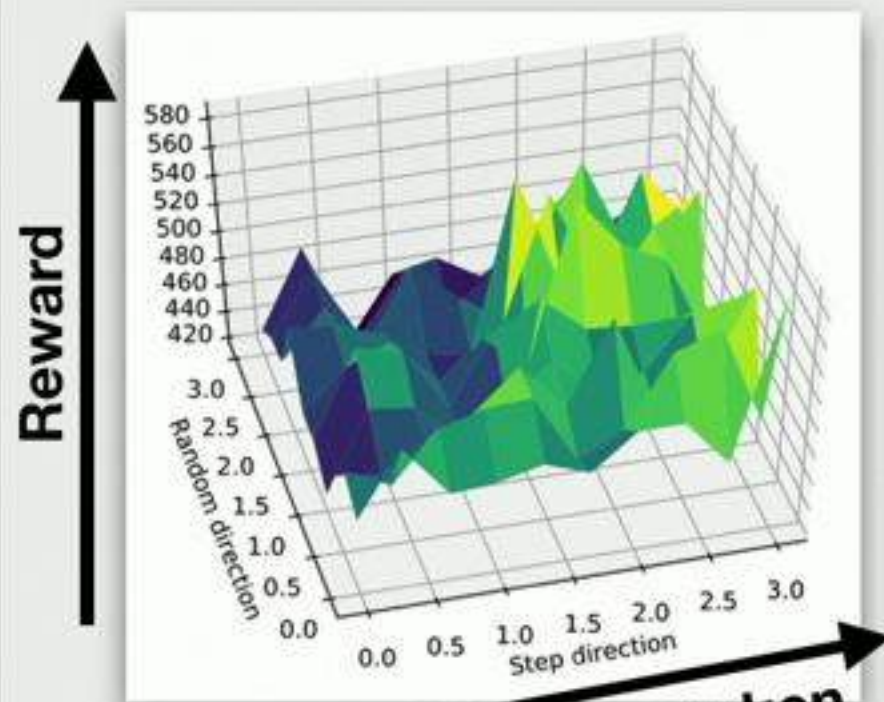
1000-sample estimates



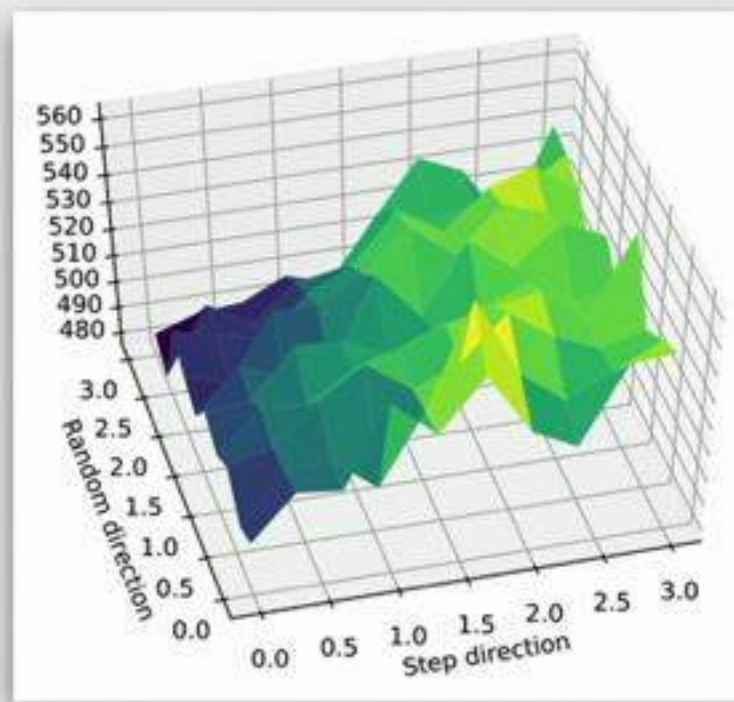
agent step taken

Optimization Landscapes

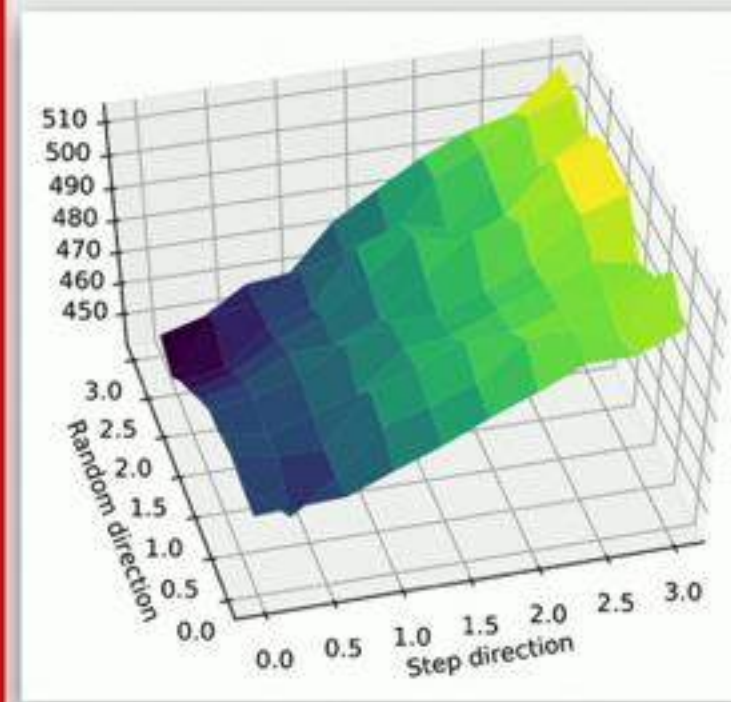
20-sample estimates



200-sample estimates

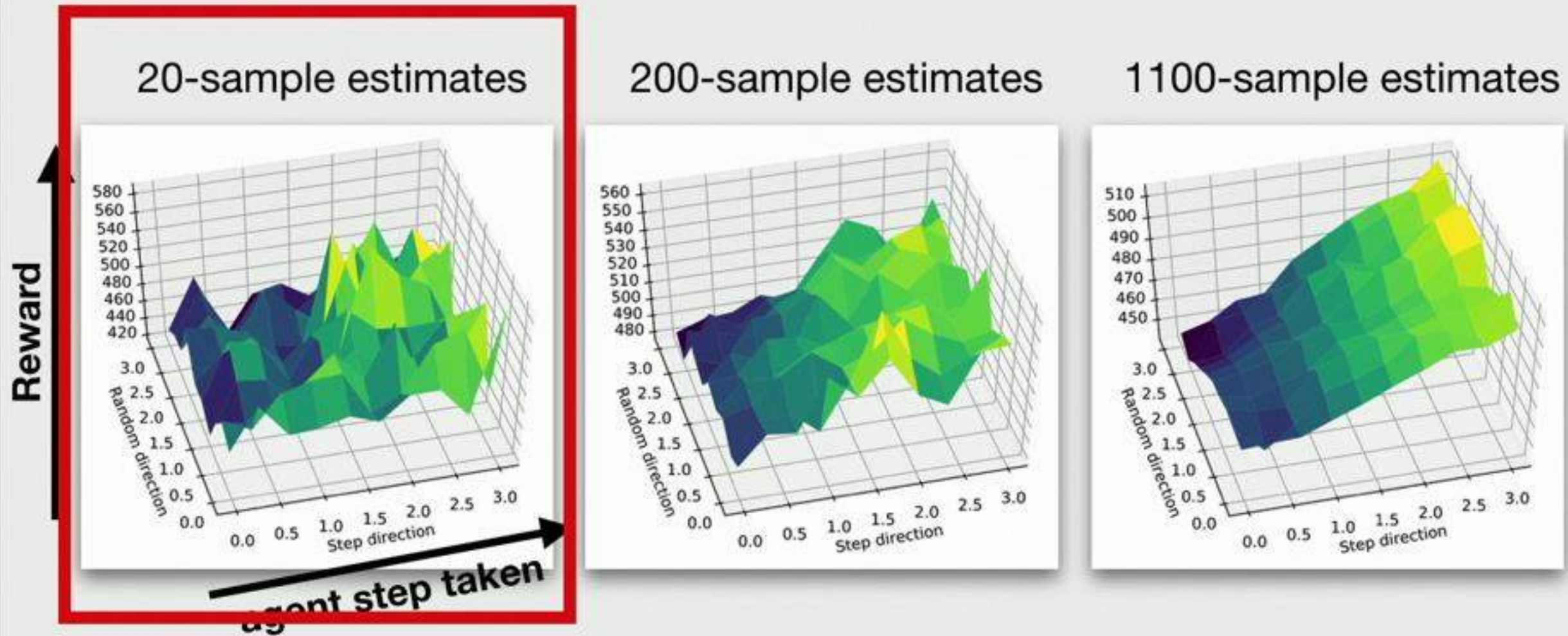


1100-sample estimates



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

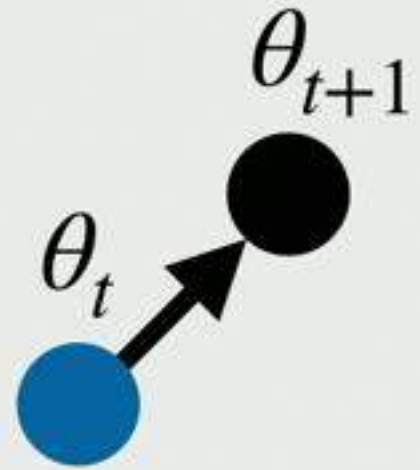
- ▶ Surrogate landscapes are often not reflective of rewards
- ▶ How can we better navigate the reward landscape?

Trust Regions

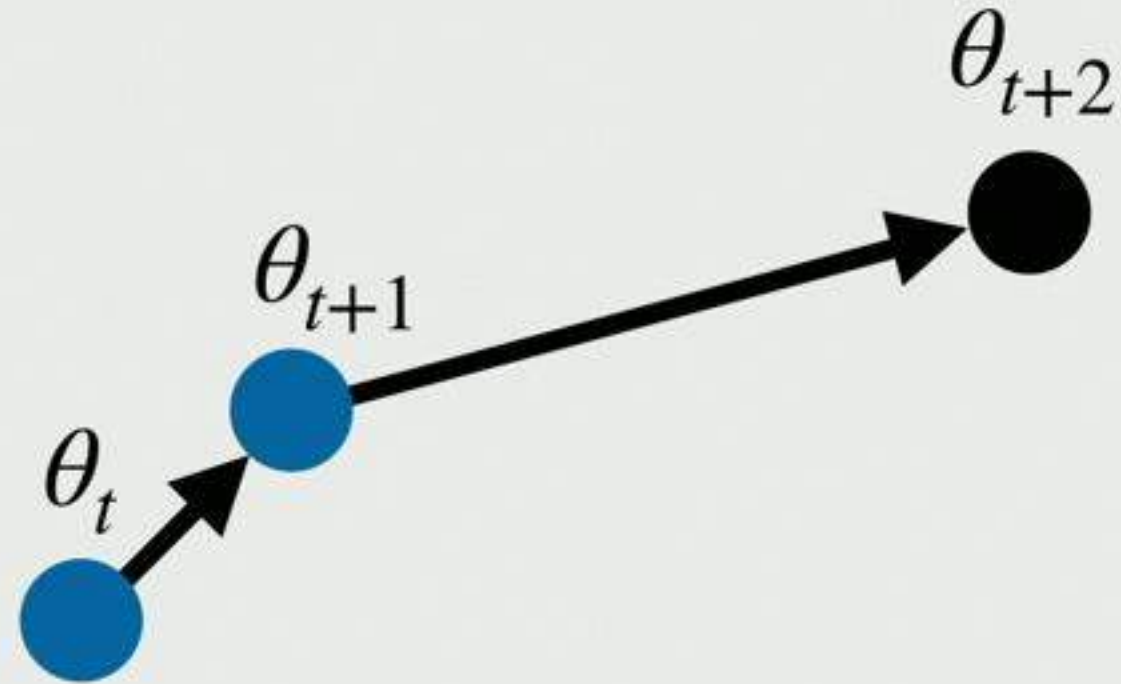
θ_t



Trust Regions



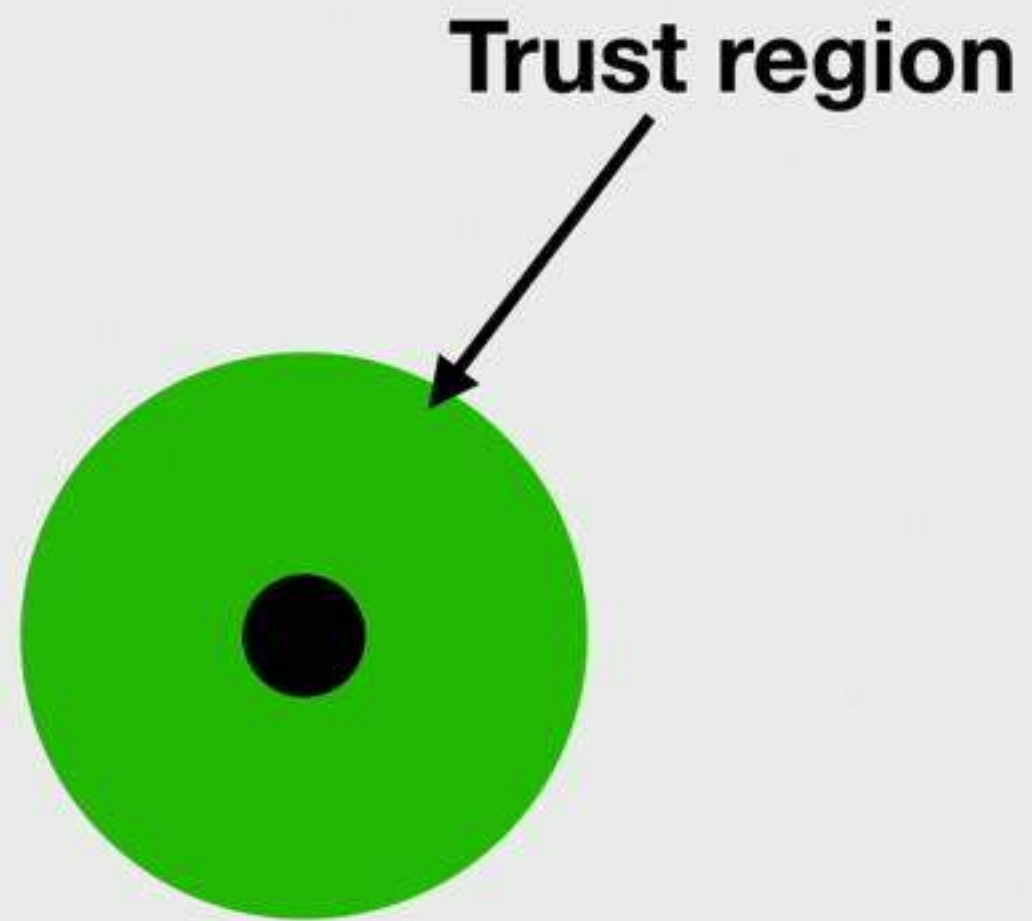
Trust Regions



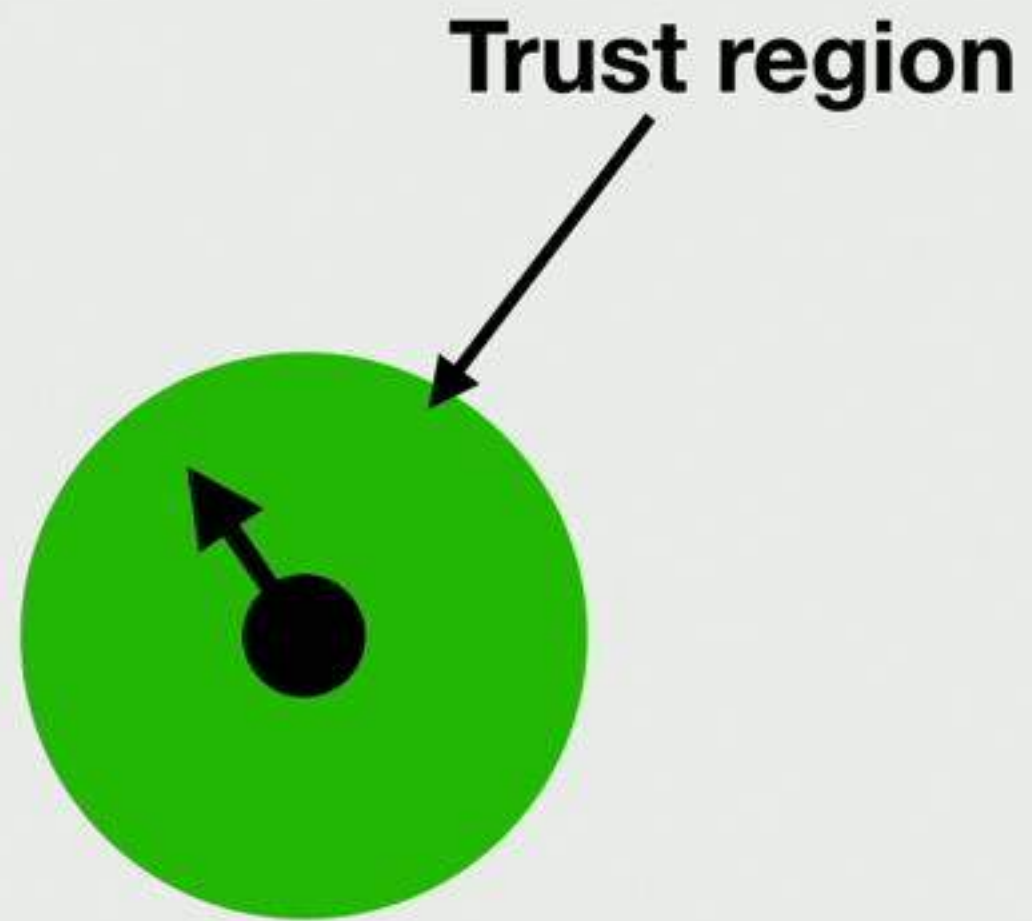
Trust Regions



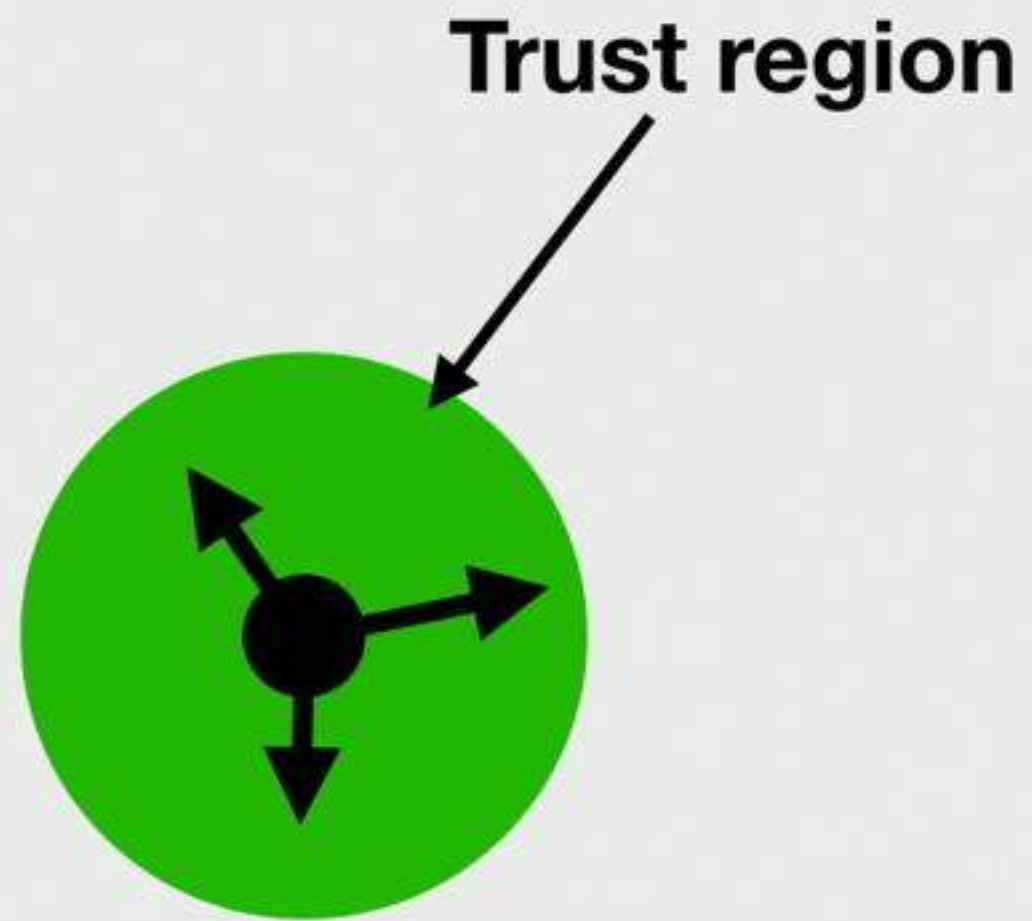
Trust Regions



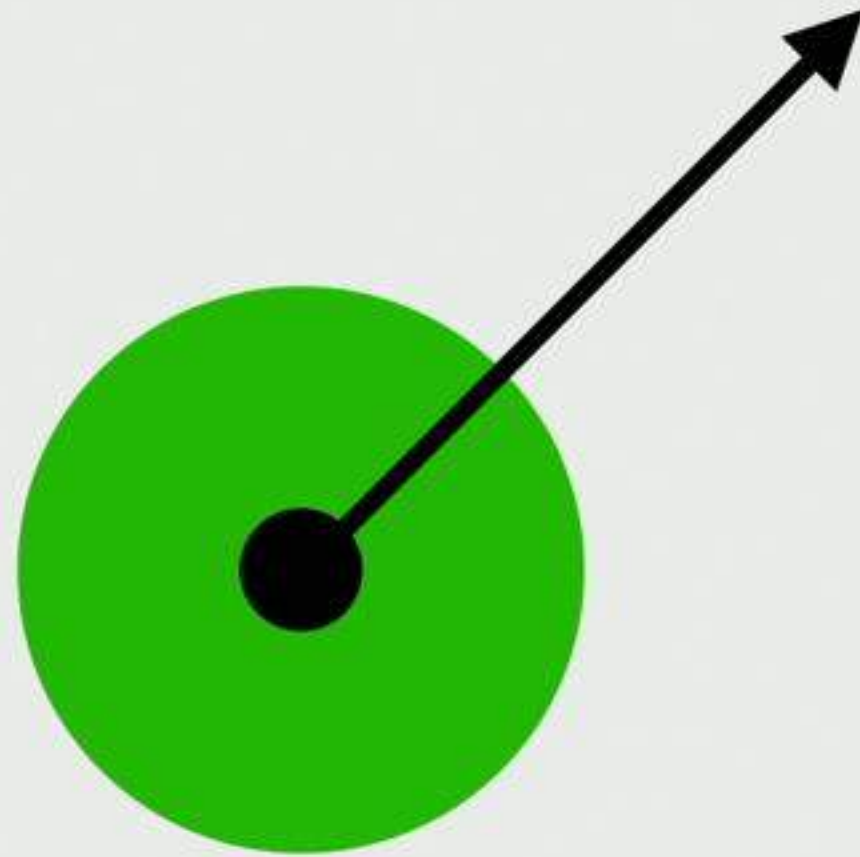
Trust Regions



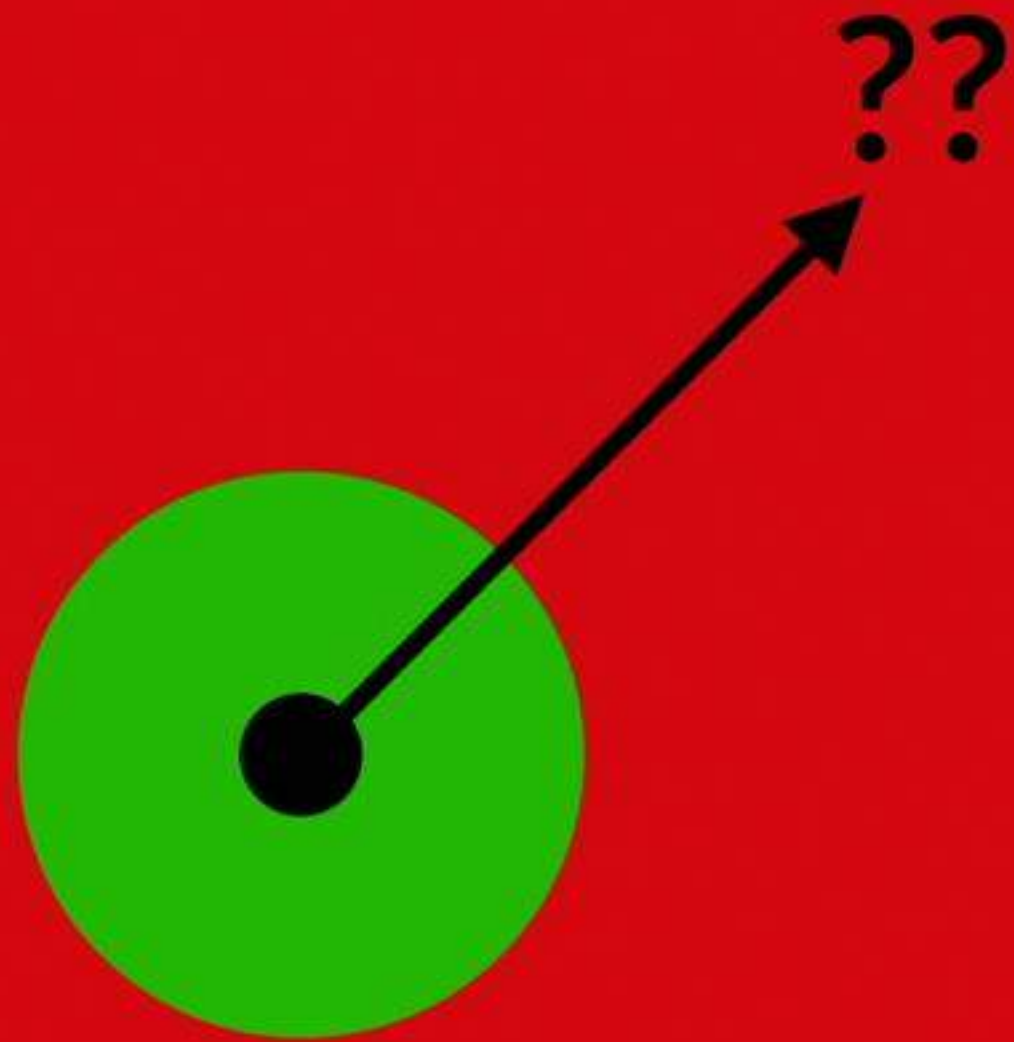
Trust Regions



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:

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

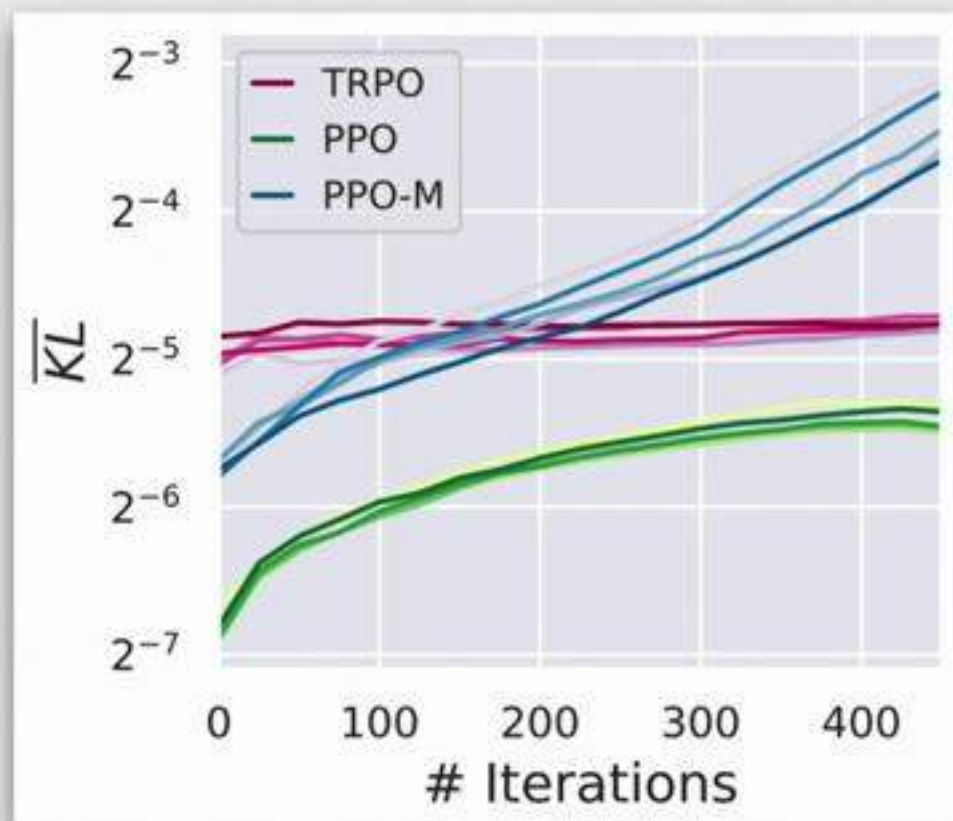
“keep the mean distance between action distributions small”

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

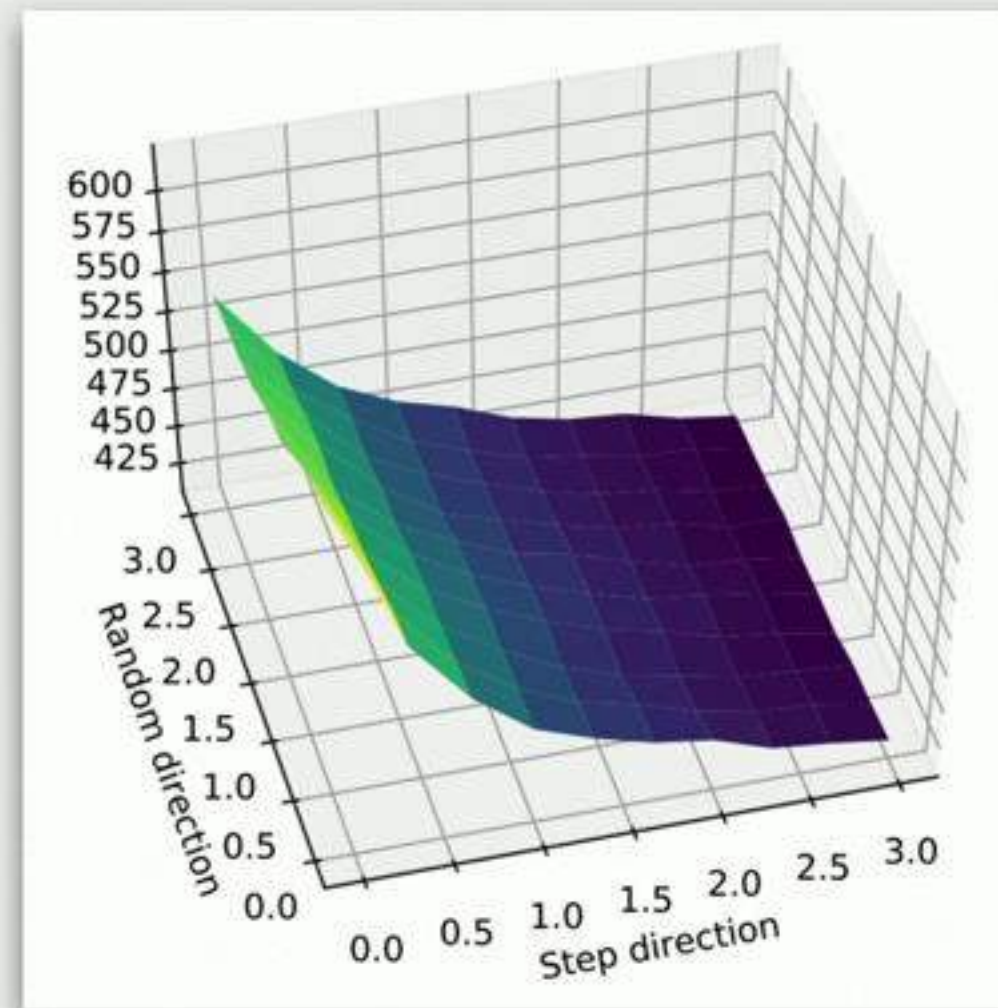
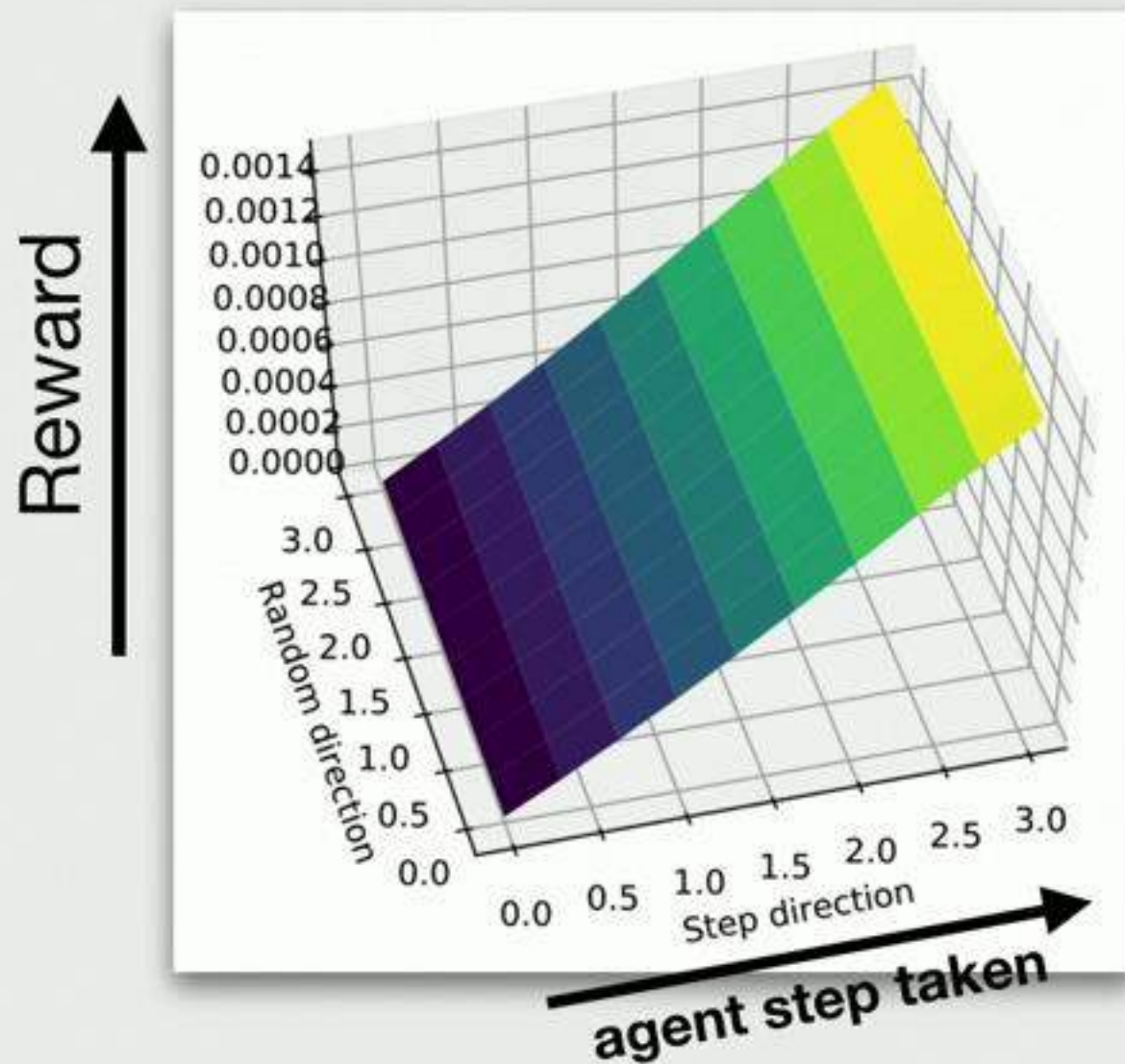


<https://bit.ly/2WVPjAn>

Optimization Landscapes

Surrogate Landscape

Reward Landscape



Step 450

Read more

Paper



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

Blog Posts



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