Some Recent Algorithmic Questions in Deep Reinforcement Learning
CS 285

Instructor: Aviral Kumar
UC Berkeley

Berkeley
UNIVERSITY OF CALIFORNIA

What Will We Discuss Today?

So far, we have gone over several interesting RL algorithms, and some theoretical aspects in RL

- Which algorithmic decisions in theory actually translate to practice, especially for Q-learning algorithms?
- Phenomena that happen in deep RL, and how we can try understanding them….
- What affects performance of various deep RL algorithms?
- Some open questions in algorithm design in **deep** RL

Disclaimer: Most material covered in this lecture is very recent and being still actively researched upon. I will present some of my perspectives on these questions in this lecture, but this is certainly not exhaustive.

Part 1: Q-Learning Algorithms

Sutton's Deadly Triad in Q-learning

$$\min_{Q} \mathbb{E}_{s,a \in \mathcal{D}} \left[\left(\left(r(s,a) + \gamma Q(s', a') \right) - Q(s,a) \right)^2 \right]$$

Boostrapping

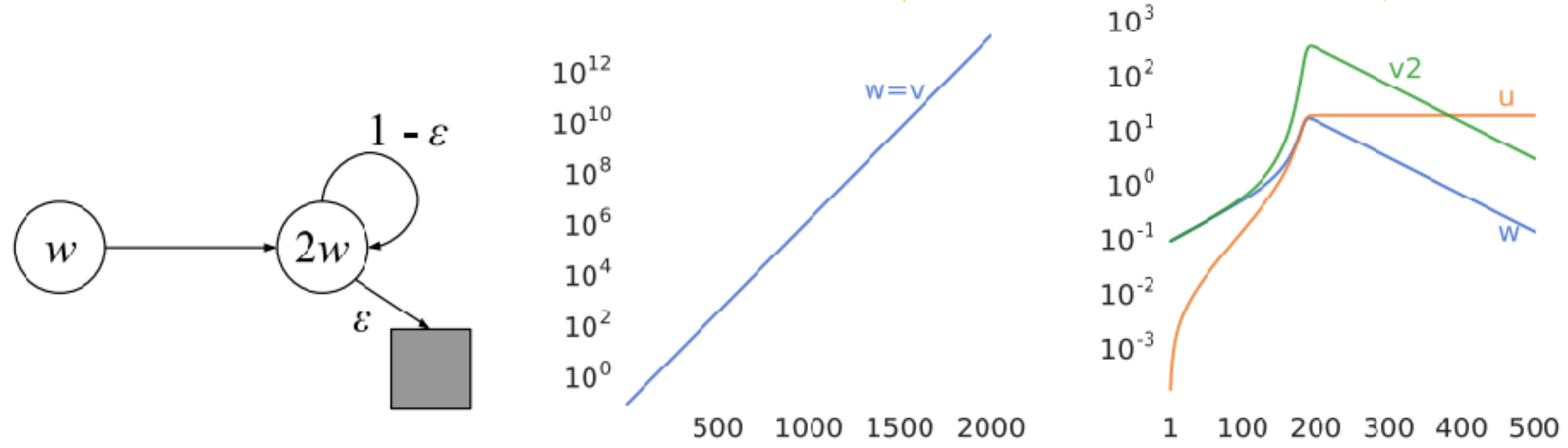
Off-policy Sampling

Function Approximation

$\Longrightarrow$ **Divergence**

Few parameters diverges

More parameters converges



(a) The example by Tsitsiklis and Van Roy (1997).

(b) $v(s) = w\phi(s)$ diverges.

(c) $v(s) = w(\phi(s) + u)$ converges.

More expressive functions approximators work fine?

Hasselt et al. Deep Reinforcement Learning and the Deadly Triad. ArXiv 2019.

What aspects will we cover?

- **Divergence:** Divergence can happen with the deadly triad and several algorithms tailored towards preventing this divergence. But does it actually happen in practice?

Large part of theory focused on fixing divergence

- **“Overfitting”/Sampling Error:** As with any learning problem, we would expect training neural network Q-learning schemes to suffer from some kind of overfitting. Do these methods suffer from any overfitting?

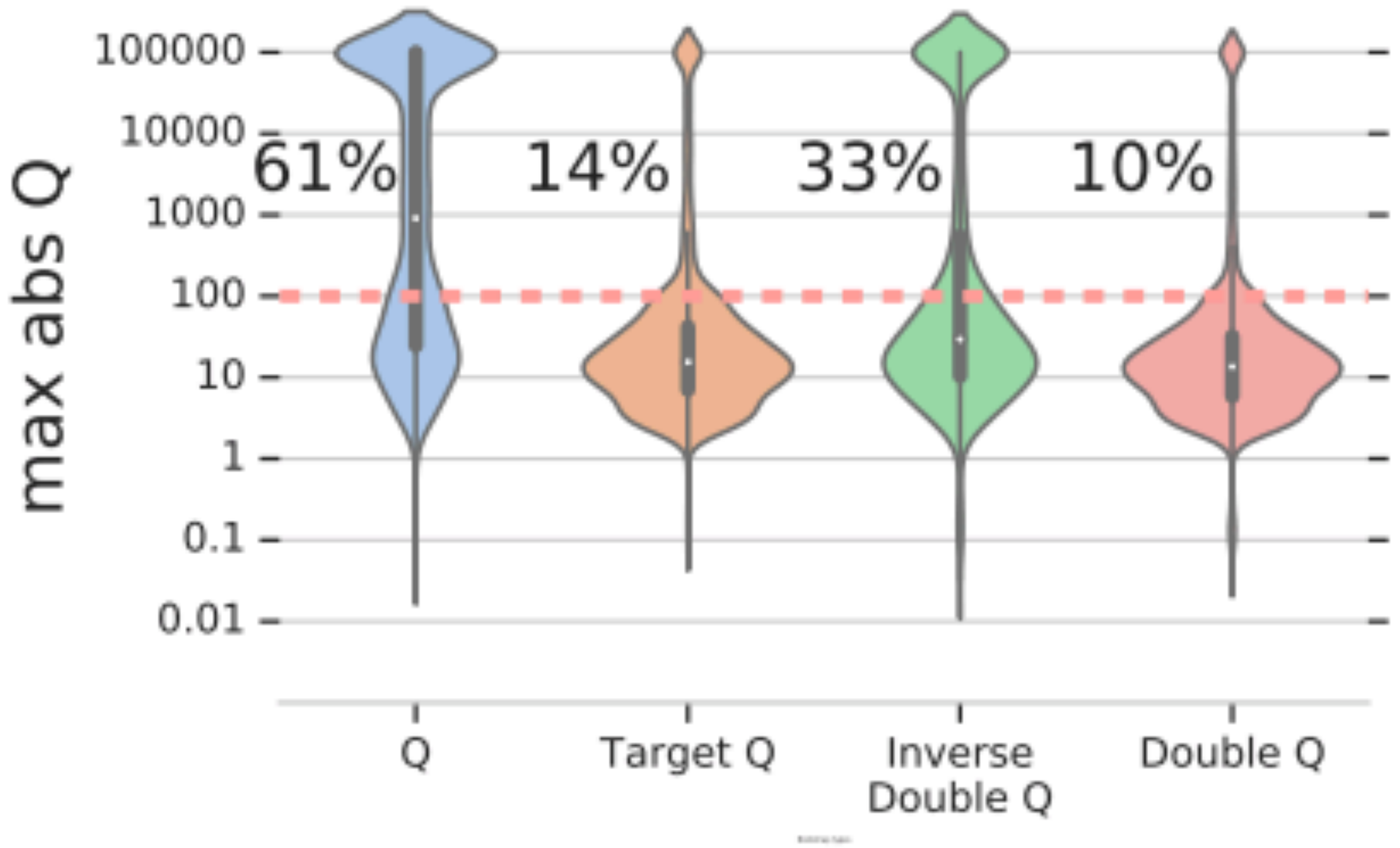
Worst-case bounds exist, but we do not know how things behave in practice

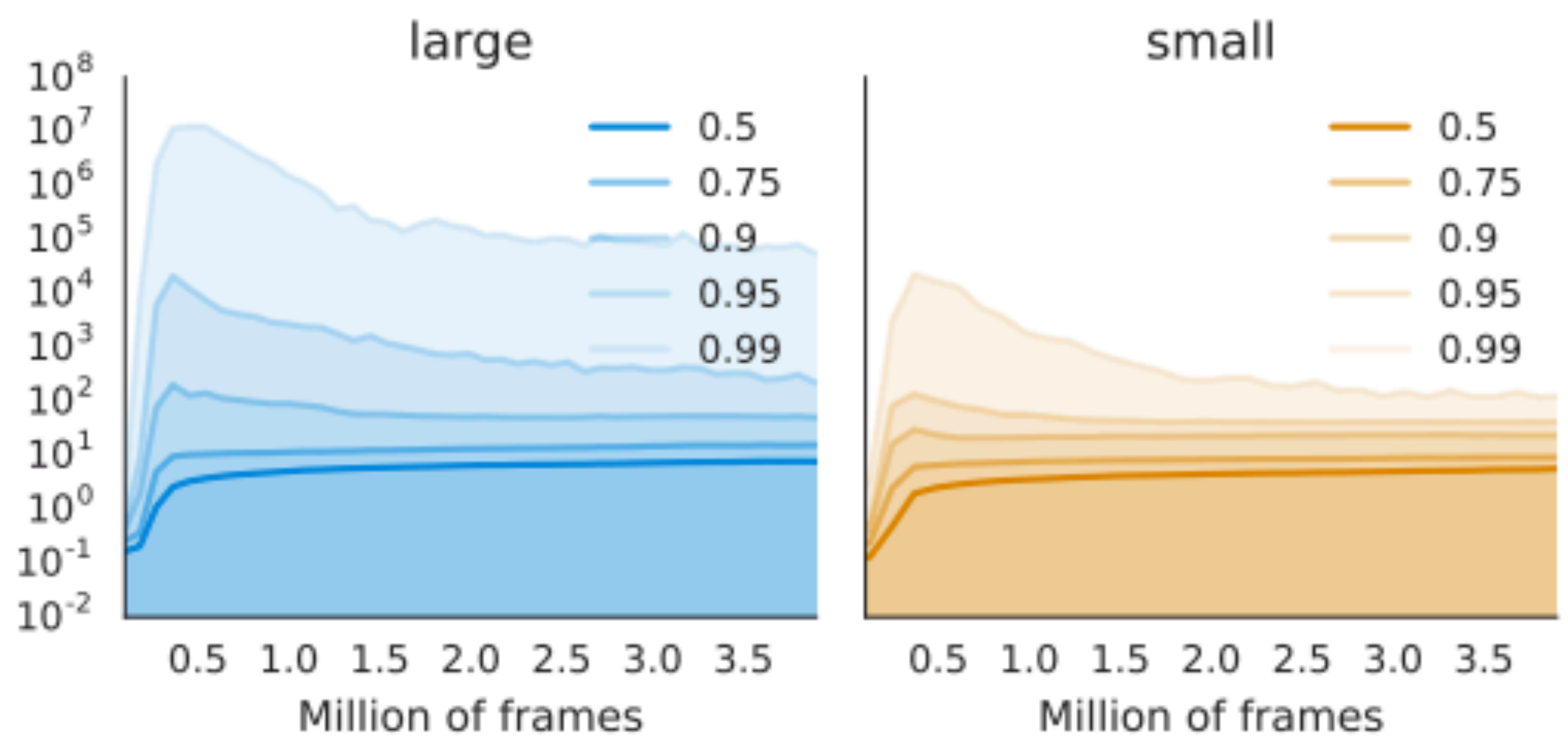
- **Data distribution:** Off-policy distributions can be bad, moreover narrow data distributions can give brittle solutions? So, which data distributions are good, and how do we get those distributions?

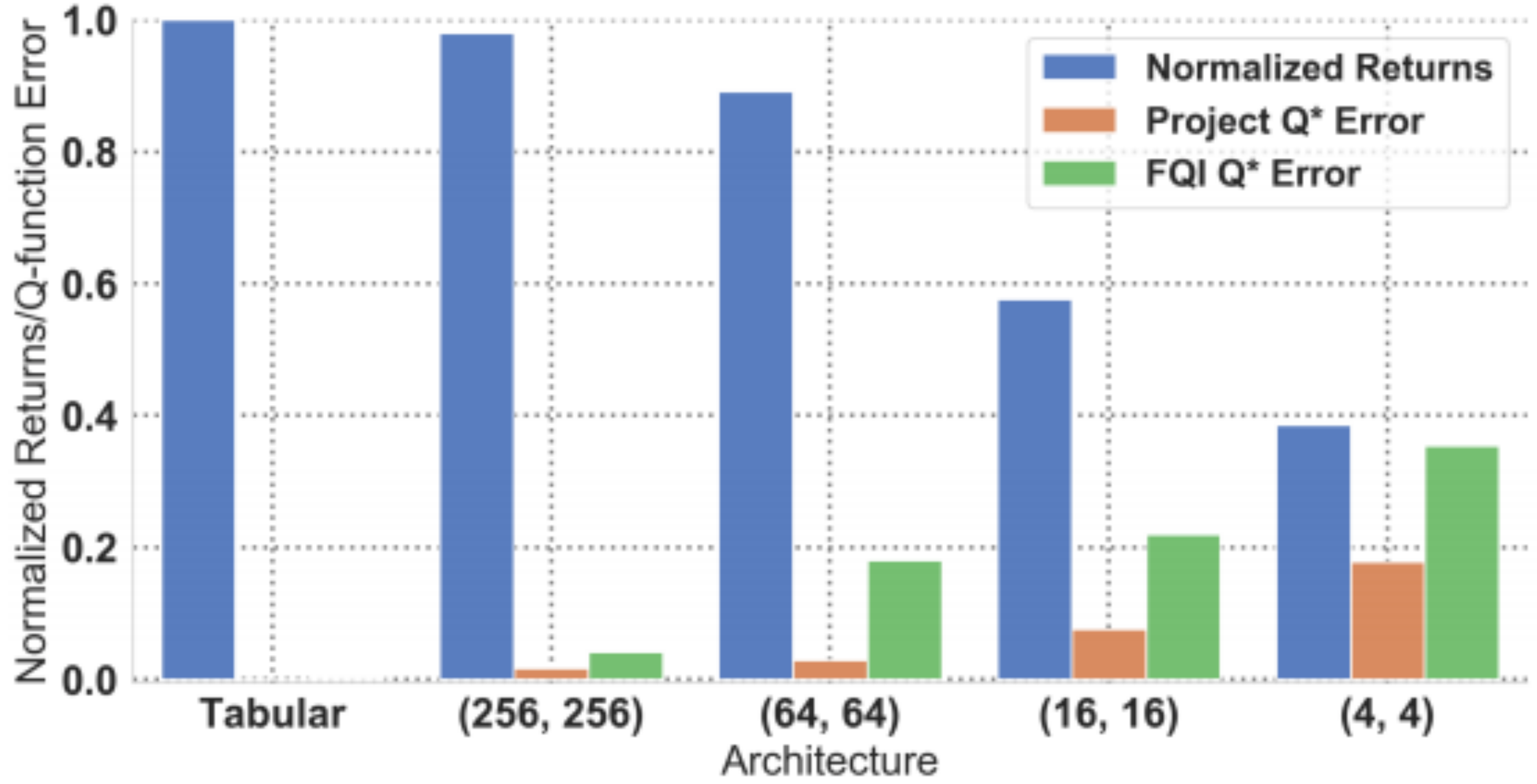
(Too) worst-case bounds, but how do things behave in practice?

Divergence in Deep Q-Learning

While Q-values are overestimated, there is not really significant divergence







Large neural networks just seem fine in an FQI-style setting

0.9% divergence

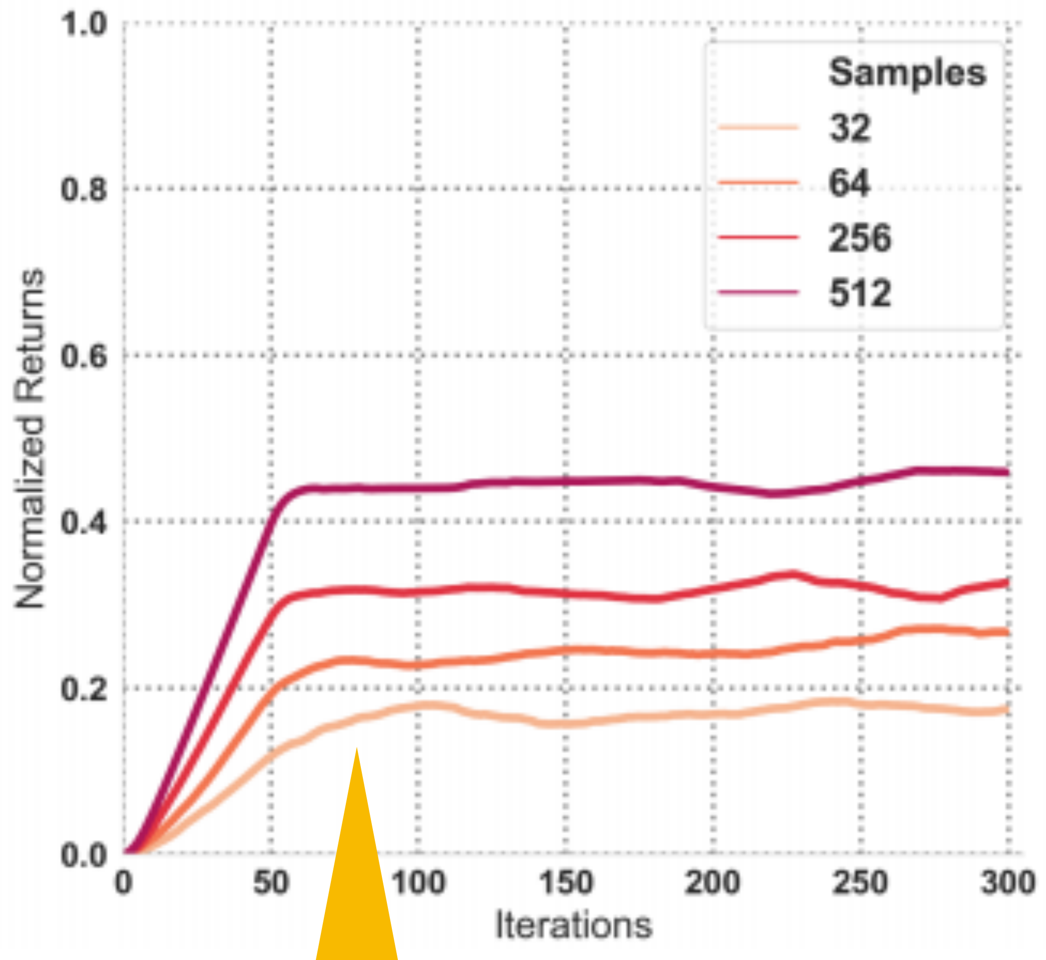
Hasselt et al. Deep Reinforcement Learning and the Deadly Triad. ArXiv 2019.

Fu*, **Kumar***, Soh, Levine. Diagnosing Bottlenecks in Deep Q-Learning Algorithms. ICML 2019

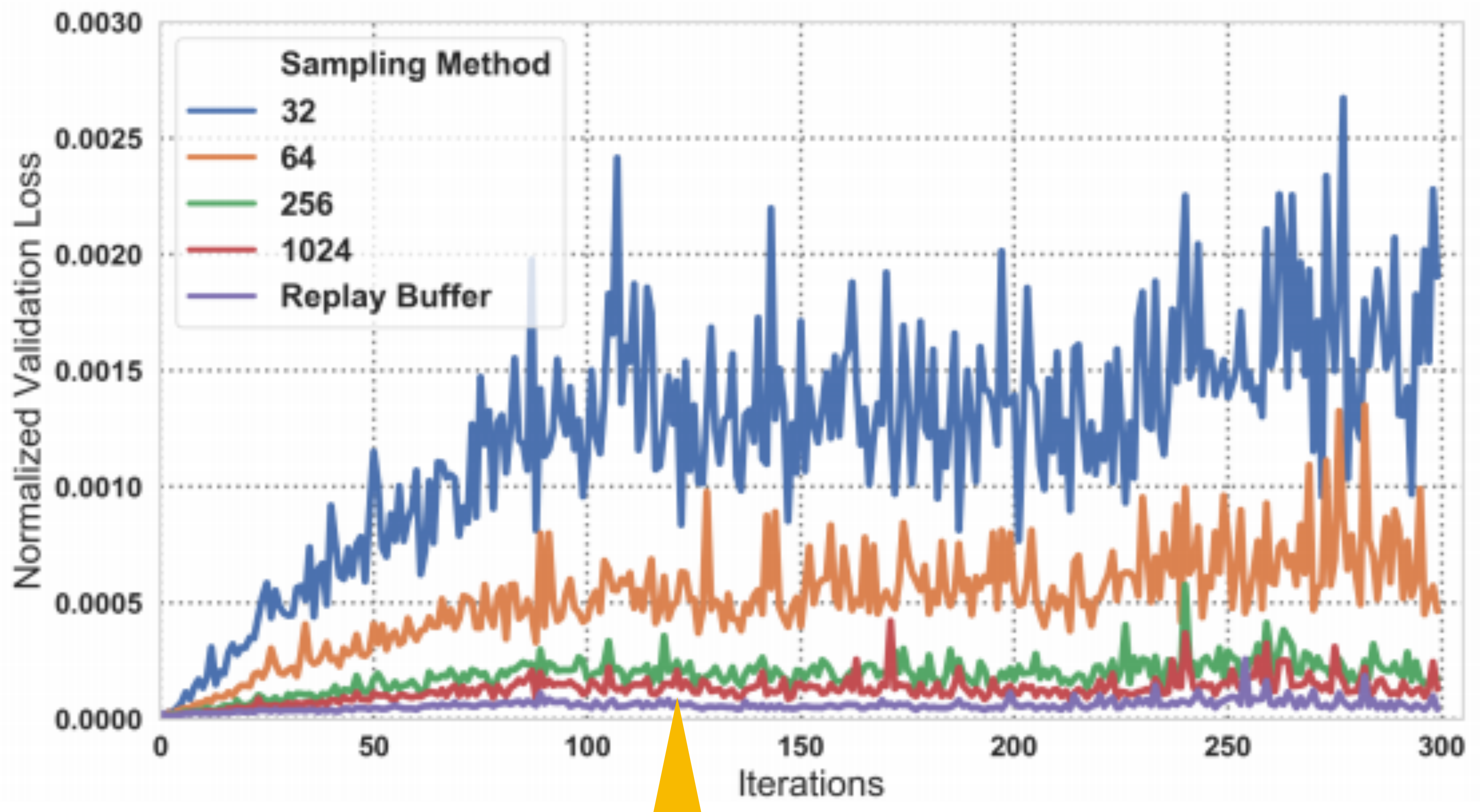
Overfitting in Deep Q-Learning

Does overfitting happen for real in FQI with neural networks?

$$\min_{Q} \quad \mathbb{E}_{s,\cancel{a}\sim\mu}\left[((r(s,a)+\gamma Q(s',a')) - Q(s,a))^2\right] \qquad \sum_{s,a\in\mathcal{D}}$$



Few samples leads to poor performance



Replay buffer prevents overfitting, even though it is off-policy

When moving from FQI to DQN/Actor-critic what happens?

Fu*, **Kumar*,** Soh, Levine. Diagnosing Bottlenecks in Deep Q-Learning Algorithms. ICML 2019

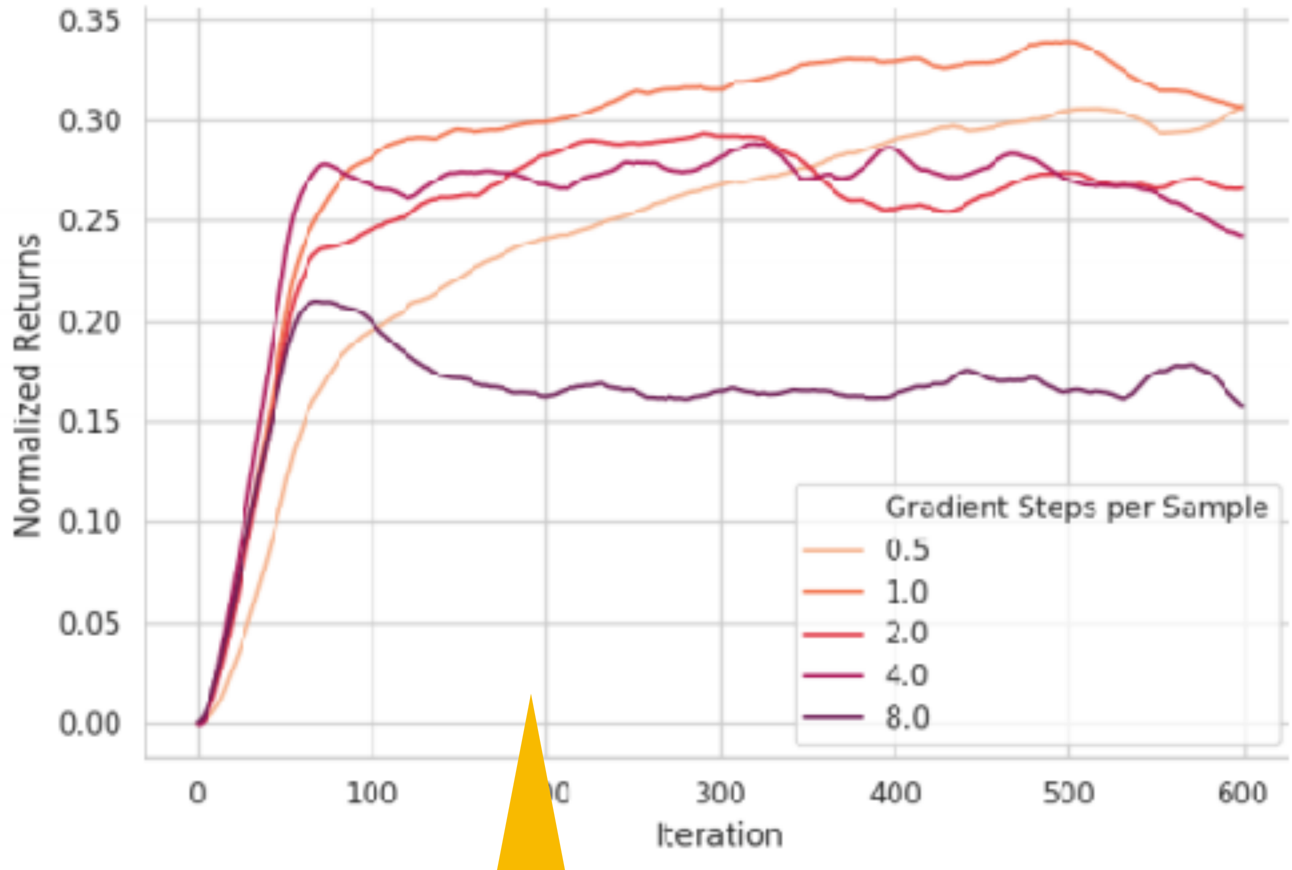
Overfitting in Deep Q-Learning

More gradient steps per environment step?

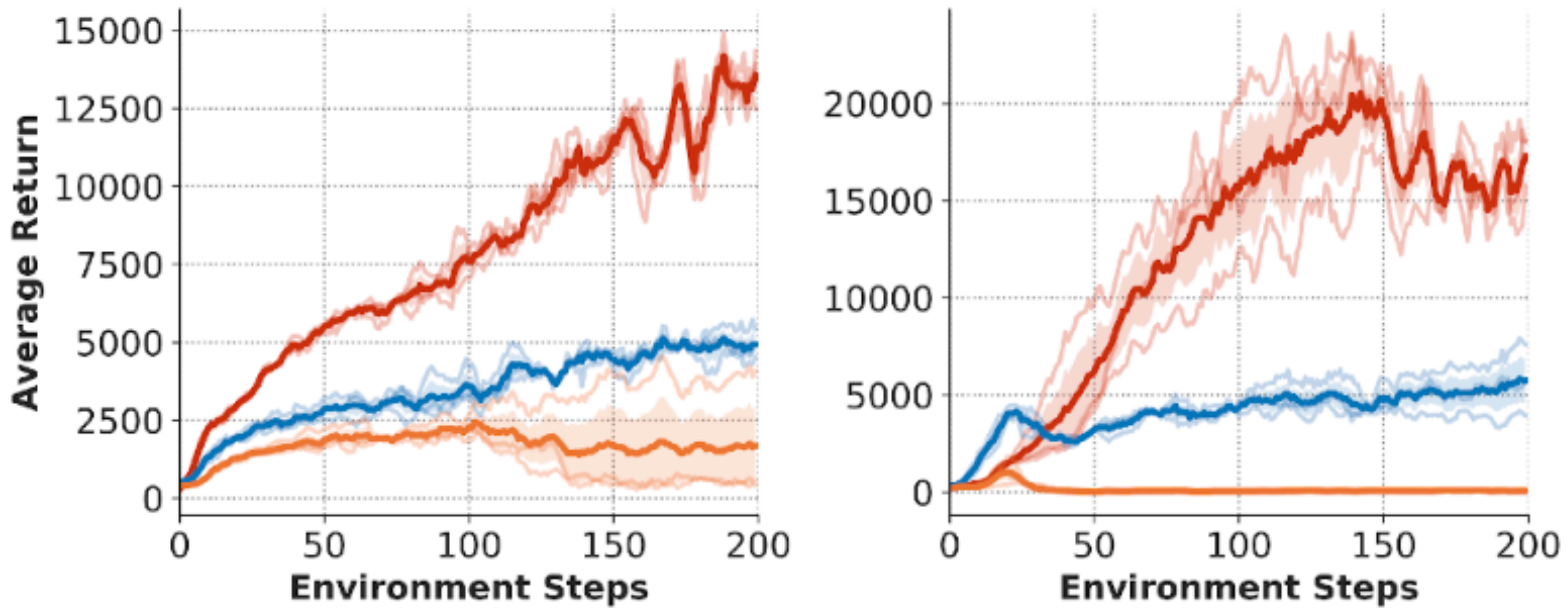
$$\frac{K}{N} = \text{gradient steps per environment step}$$

1. Sample N samples from the environment

2. Train for K steps before sampling next



More gradient steps hurt performance

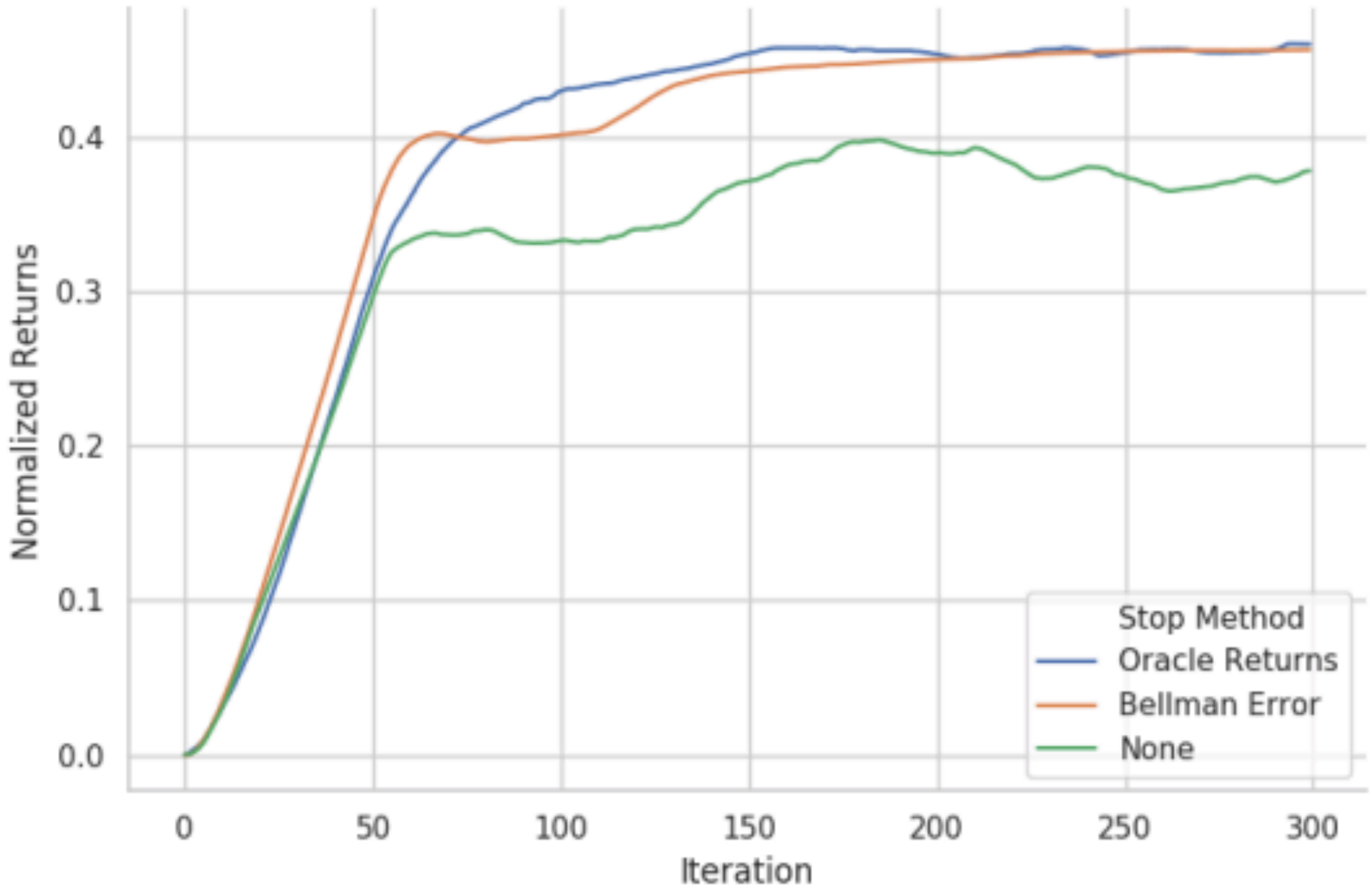


Fu*, **Kumar*,** Soh, Levine. Diagnosing Bottlenecks in Deep Q-Learning Algorithms. ICML 2019

Overfitting in Deep Q-Learning

Why does performance degrade with more training?

- **Possibility 1:** Large deep networks overfit, and that can cause poor performance — so more training leads to worse performance…
- **Possibility 2:** Is there something else with the deep Q-learning update?



Early stopping helps

Although this is with "oracle access" to Bellman error on all states…. so not practical

Fu*, **Kumar*,** Soh, Levine. Diagnosing Bottlenecks in Deep Q-Learning Algorithms. ICML 2019

Overfitting in Deep Q-Learning

- **Possibility 2 is also a major contributor:** Performance often depends on the fact that optimization uses a bootstrapped objective — i.e. uses labels from itself for training

Self-creating labels for training can hurt

Preliminaries: Gradient descent with deep networks has an implicit regularisation effect in supervised learning, i.e. it regularizes the solution in overparameterized settings.

$$\min_{X} \quad ||AX - y||_2^2$$

If gradient descent converges to a good solution, it converges to a <u>minimum norm</u> solution

$$\min_{X} ||X||_F^2 \text{ s.t. } AX = y$$

Check Arora et al. (2019) for a discussion of how this regularization is more complex…

Gunasekar et al. Implicit Regularization in Matrix Factorization. NeurIPS 2017.
Arora et al. Implicit Regularization in Deep Matrix Factorization. NeurIPS 2019.
Mobahi et al. Self-Distillation Amplifies Regularization in Hilbert Space. NeurIPS 2020.

Implicit Under-Parameterization

When training Q-functions with bootstrapping on the same dataset, more gradient steps lead to a loss of expressivity due to excessive regularization, that manifests as a loss of rank of the feature matrix.

$$Q(\mathbf{s}, \mathbf{a}) = \mathbf{w}^T \Phi(\mathbf{s}, \mathbf{a}) \qquad \Phi \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times d}. \qquad \mathbf{w} \in \mathbb{R}^d$$

Learned by a neural network

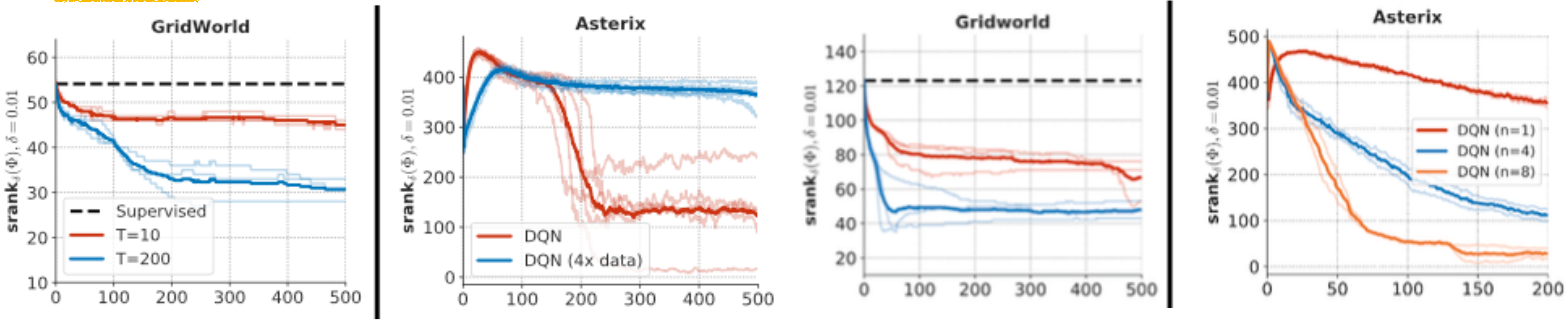
$$\Phi = U \mathrm{diag}\{\sigma_i(\Phi)\} V^T$$

$$\mathrm{srank}_\delta(\Phi) = \min \left\{ k : \frac{\sum_{i=1}^{k} \sigma_i(\Phi)}{\sum_{i=1}^{d} \sigma_i(\Phi)} \geq 1 - \delta \right\}$$

Effective rank

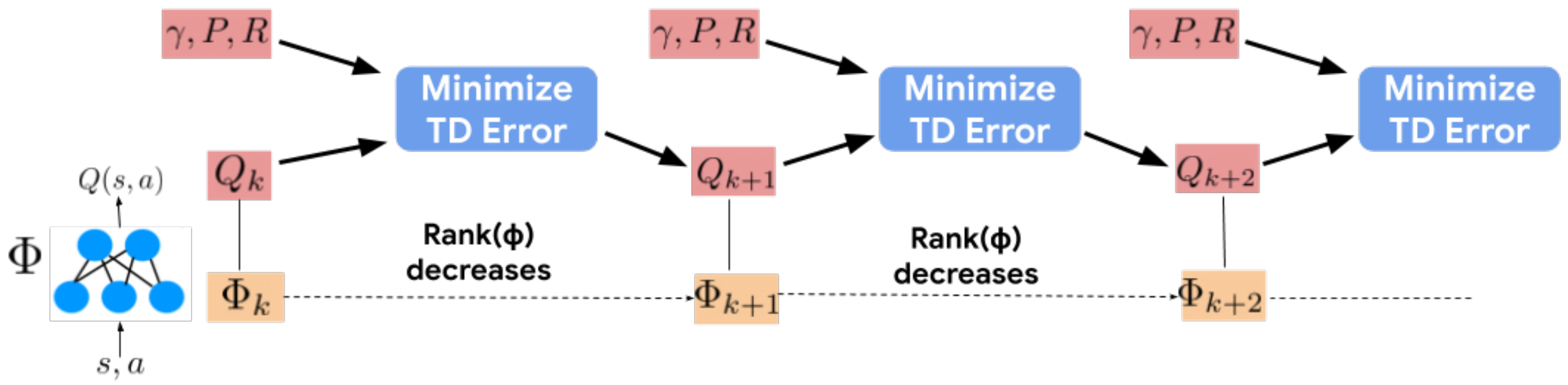
Offline

Online



Kumar*, Agarwal*, Ghosh, Levine. Implicit Under-Parameterization Inhibits Data-Efficient Deep RL. 2020.

Implicit Under-Parameterization

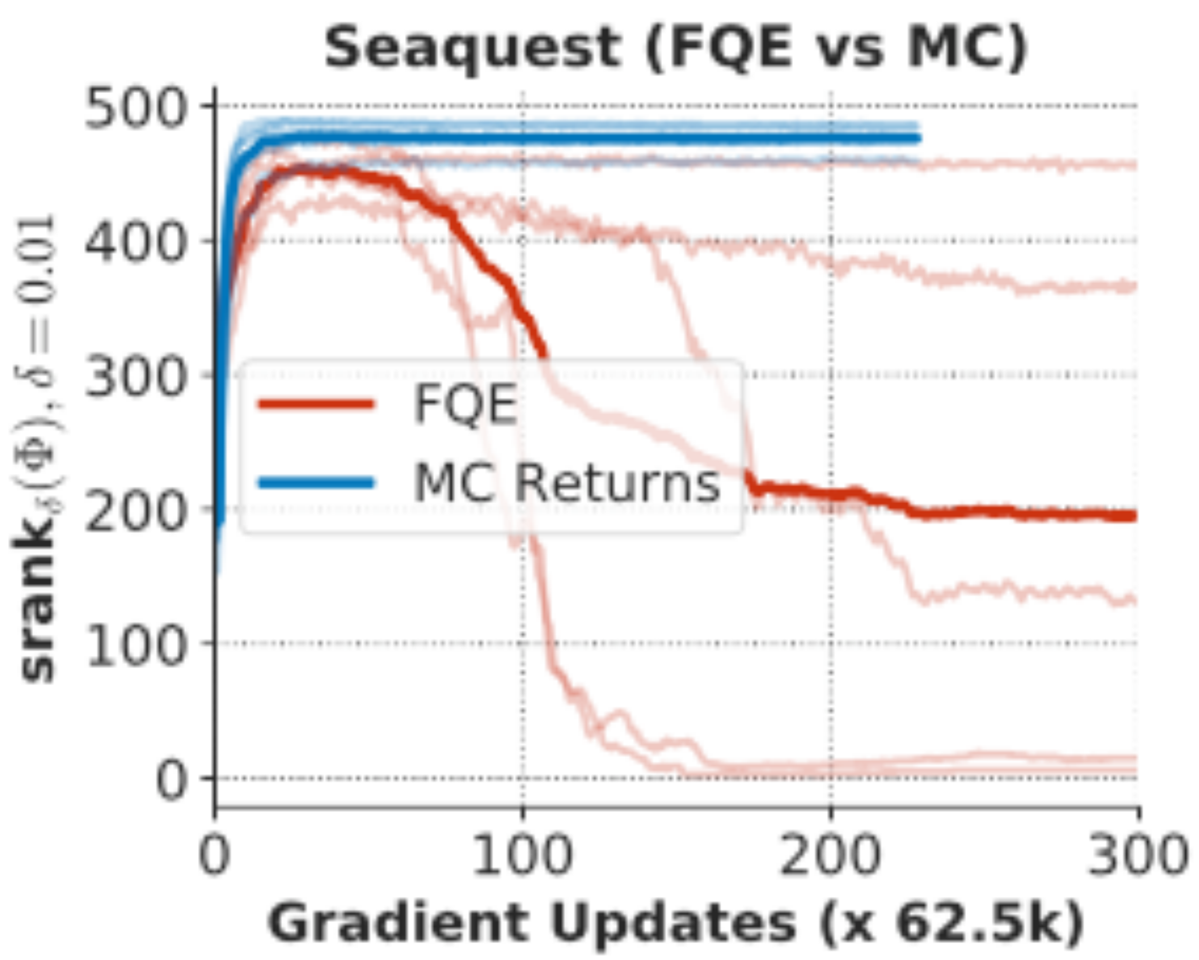


Compounding effect of rank drop over time, since we regress to labels generated from our own previous instances (boostrapping)

Kumar*, Agarwal*, Ghosh, Levine. Implicit Under-Parameterization Inhibits Data-Efficient Deep RL. 2020.

Implicit Under-Parameterization

Does implicit under-parameterization happen due to bootstrapping?



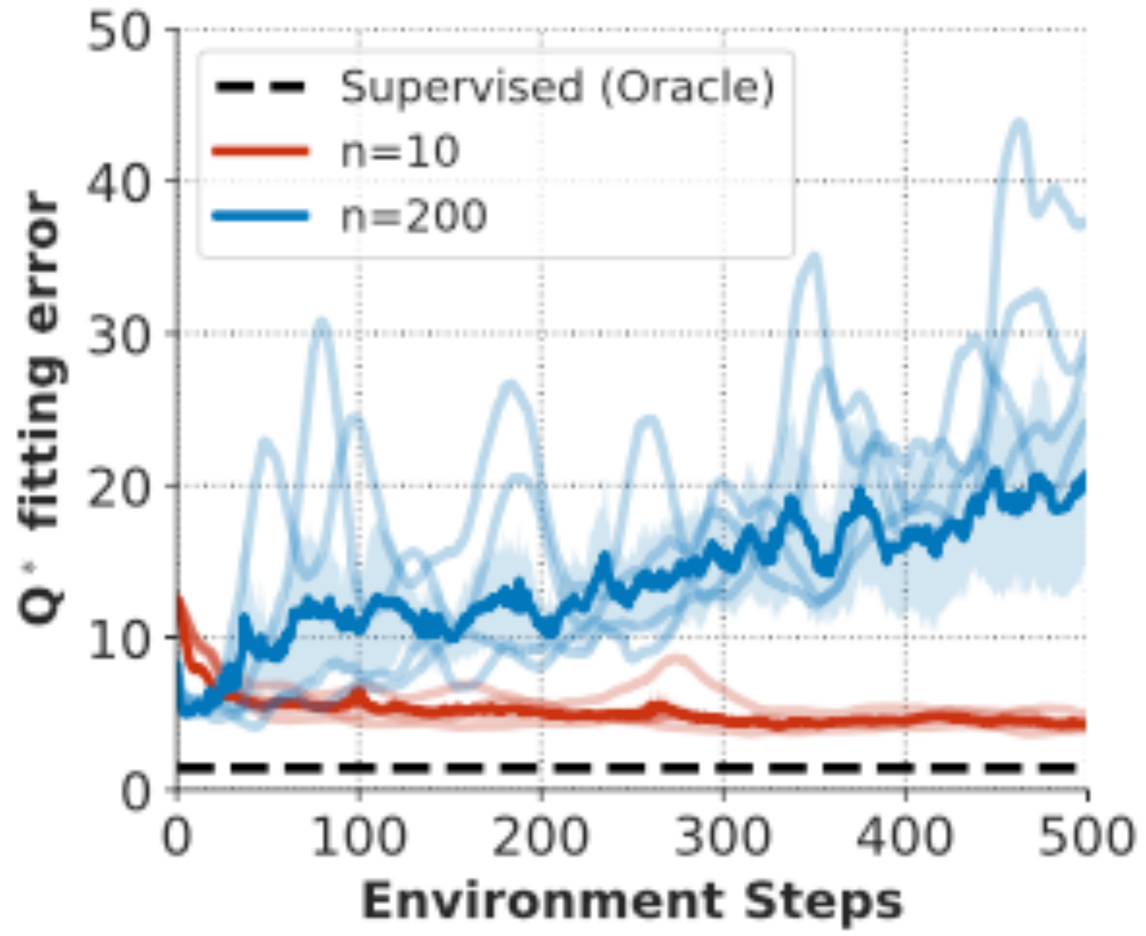
Doesn't happen when bootstrapping is absent

$$Q(s,a) = r(s,a) + \gamma E_{s',a' \sim P(s'|s,a)\pi(a'|s')}[Q(s',a')]$$

$$Q(s_0,a_0) = \sum_{t=0}^{\infty} \gamma^t r_t(s_t,a_t)$$

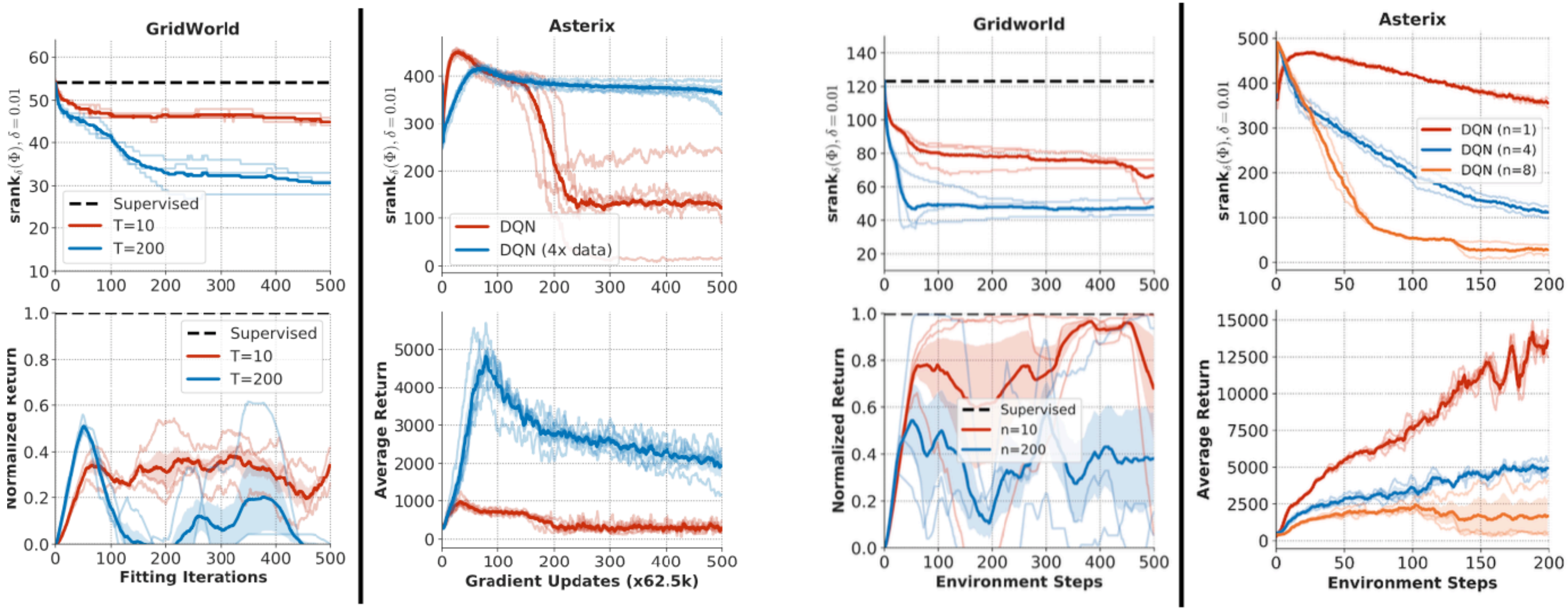
It hurts the representability of the optimal Q-function

On the gridworld example from before, representing the optimal Q-function becomes hard with more rank drop!



Effective Rank and Performance

Rank collapse corresponds to poor performance



Also observed on the gym environments, rank collapse corresponds…

Kumar*, Agarwal*, Ghosh, Levine. Implicit Under-Parameterization Inhibits Data-Efficient Deep RL. 2020.

Data Distributions in Q-Learning

- Deadly triad suggests poor performance due to off-policy distributions, bootstrapping and function approximation.
- Are on-policy distributions much better for Q-learning algorithms?
- If not, then what factor decides which distributions are "good" for deep Q-learning algorithms?

Experimental Setup

$$\min_{Q} \mathbb{E}_{s,a\sim p}\left[\left(Q(s,a) - (r(s,a) + \gamma \max_{a'} \bar{Q}(s',a'))\right)^2\right]$$

$$\mathcal{H}(p) = -\sum_{(s,a)} p(s,a) \log p(s,a)$$

Measures the entropy/ uniformity of weights

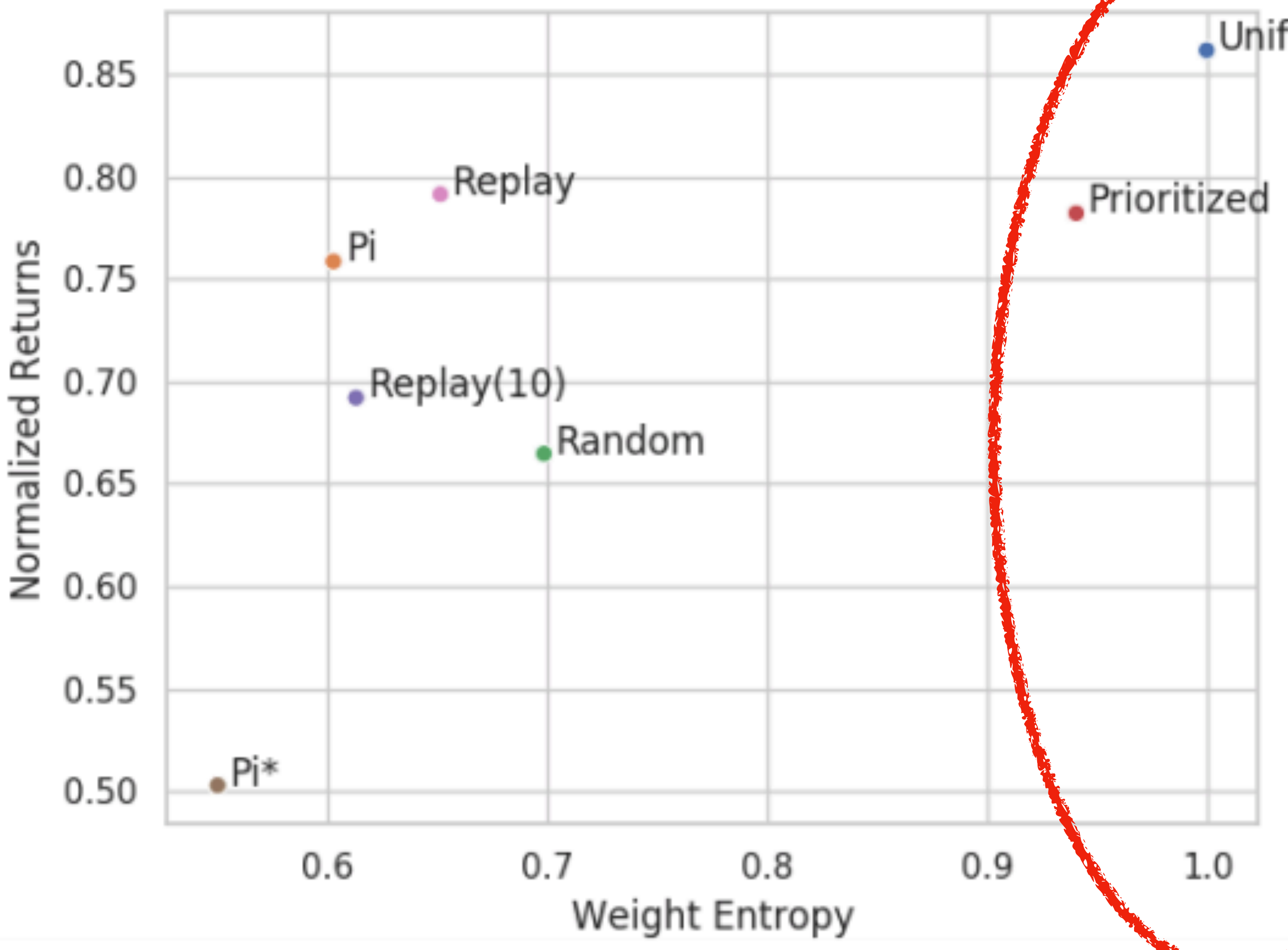
Which Data-Distributions are Good?

Compare different data distributions:

Replay | On-policy (Pi) | Optimal Policy | Uniform over (s, a)

$$\min_{Q} \max_{p} \mathbb{E}_{s,a \sim p} \left[\left(Q(s,a) - T\bar{Q}(s,a) \right)^2 \right]$$

Prioritized



$\mathcal{H}(p)$

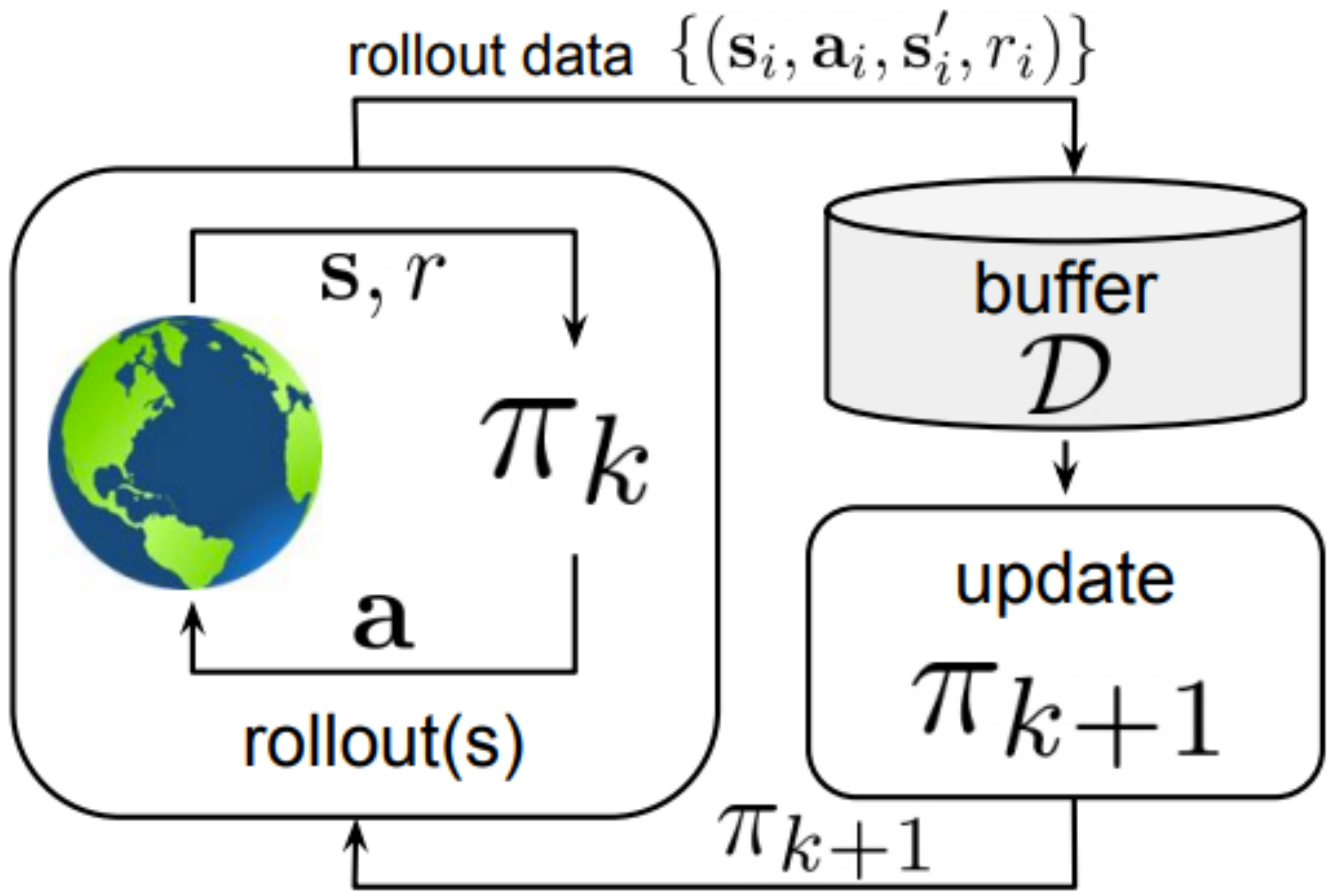
High entropy weights are good for performance

No sampling error here, all state-action pairs provided to the algorithm

Do replay buffers work because of more coverage? Maybe…

Not always, lead to biased training

Finding Good Data-Distributions



Corrective feedback = the ability of data collection to correct errors in the Q-function.

$$|Q_k(s, a) - Q^*(s, a)|$$

Function Approximation

On-policy data collection

"Corrective Feedback"

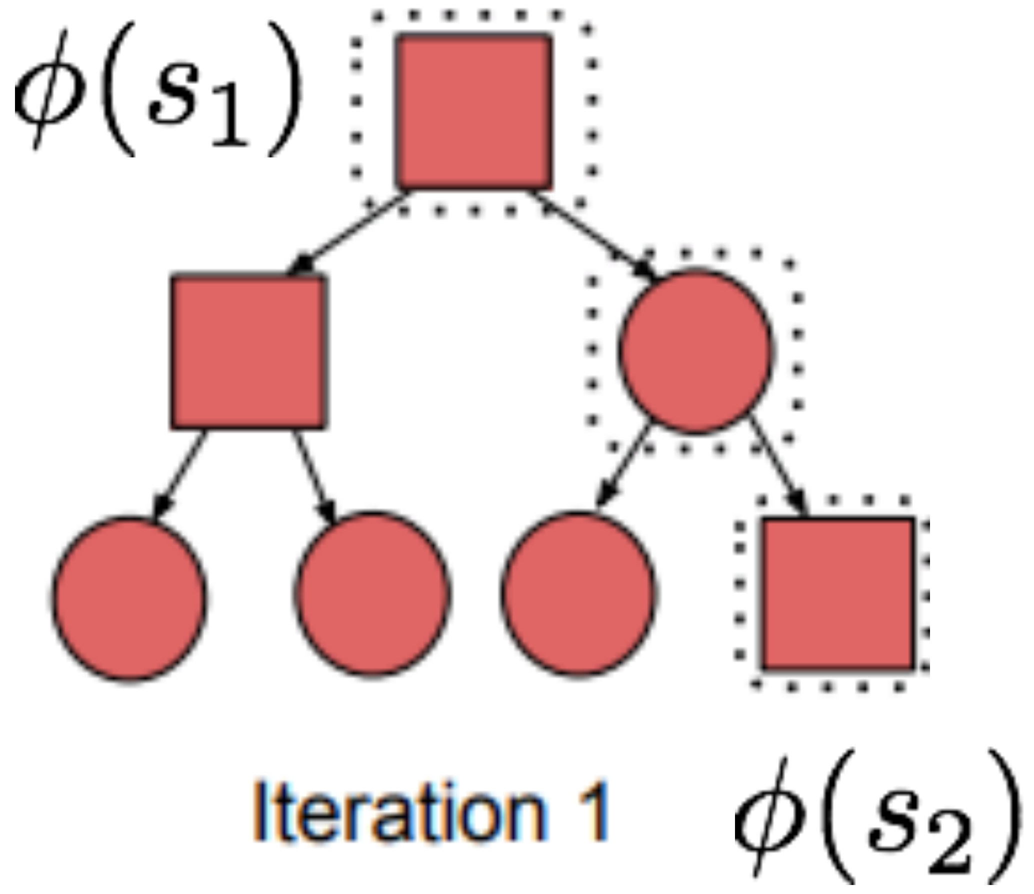
Includes replay buffer distributions

What we'll show is that on-policy data collection may fail to correct errors in the target values that are important to be backed up..

Kumar, Gupta, Levine. DisCor: Corrective Feedback in Reinforcement Learning via Distribution Correction. NeurIPS 2020.

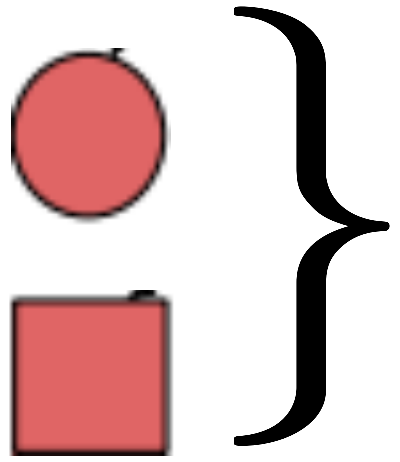
Consider This Example…

- Let’s start with a simple case of an MDP with function approximation

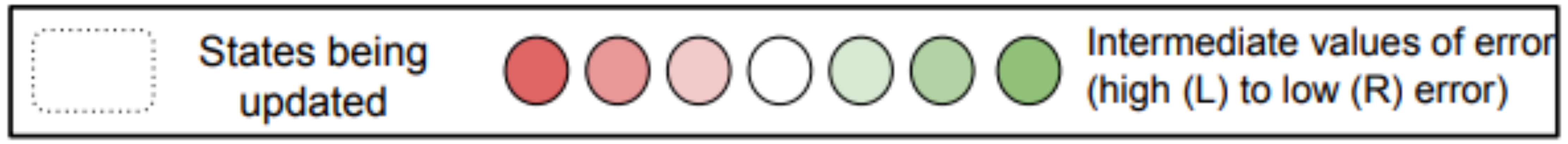


$\phi(s_1)$ and $\phi(s_2)$ are related .

$$Q(s,a) = w_a^T \phi(s)$$



Nodes are aliased with other nodes of the same shape

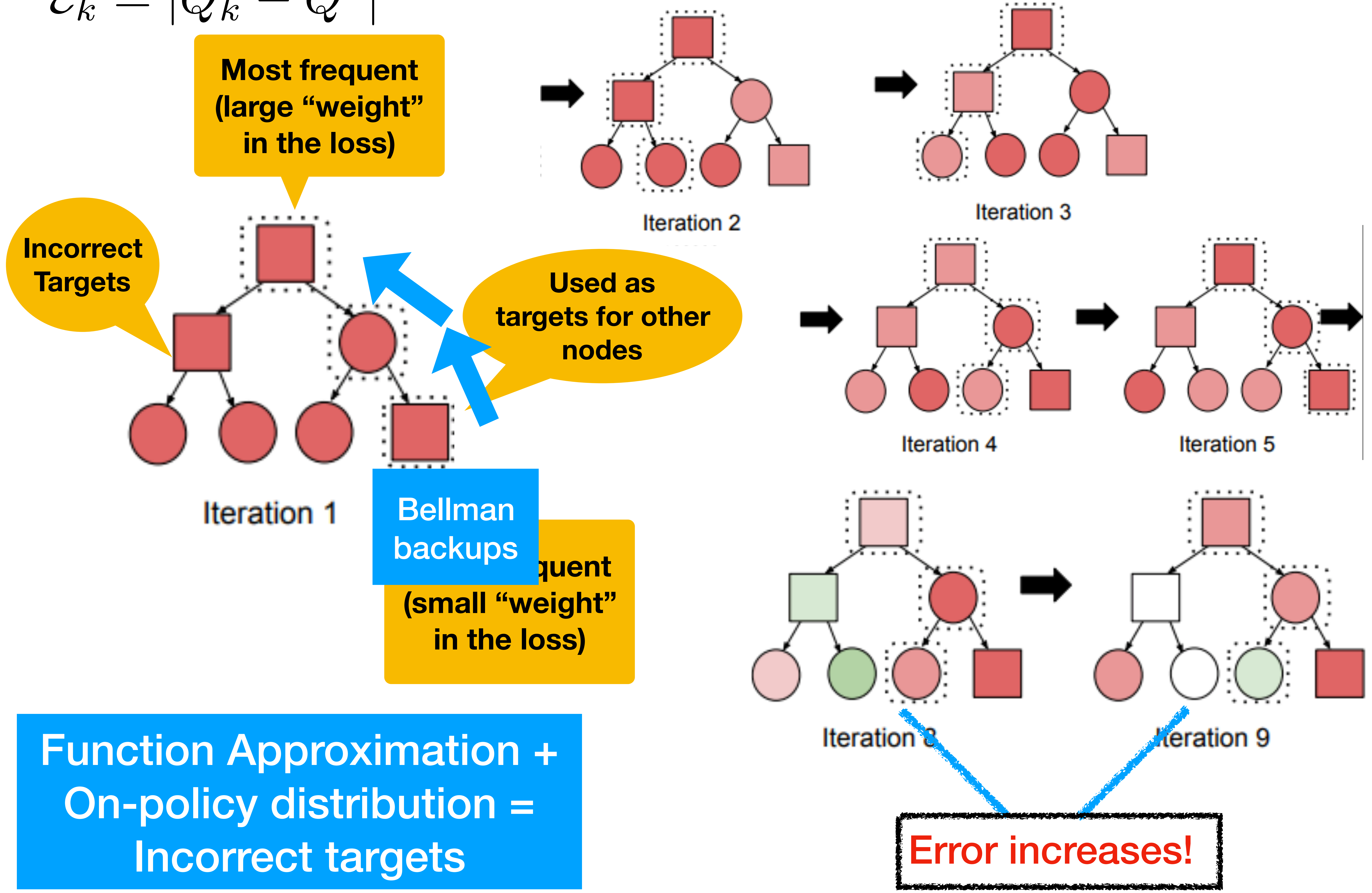


Data distribution would affect solutions in the presence of aliasing

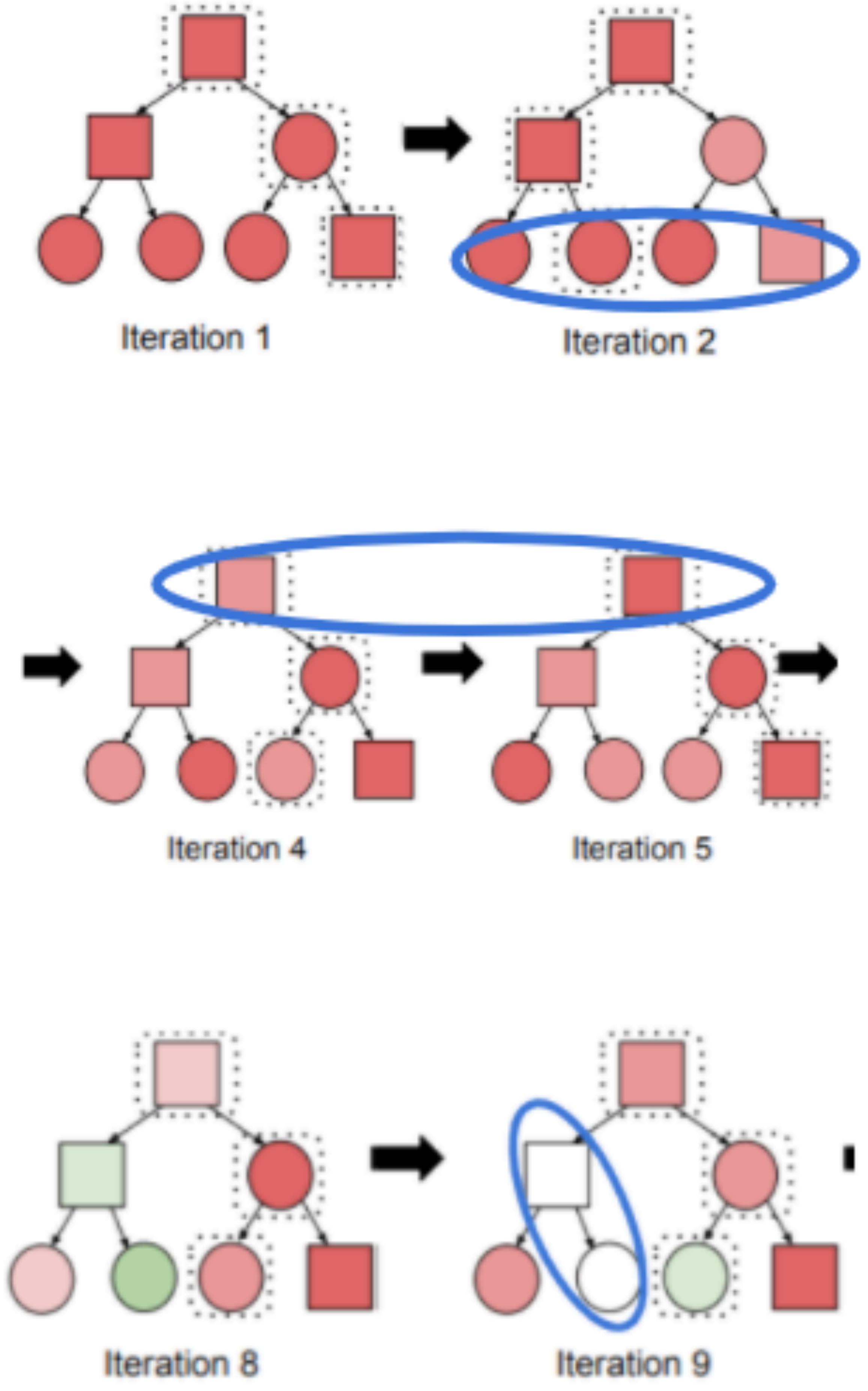
Kumar, Gupta, Levine. DisCor: Corrective Feedback in Reinforcement Learning via Distribution Correction. NeurIPS 2020.

Q-Learning with On-Policy Data Collection

$$\mathcal{E}_k = |Q_k - Q^*|$$



Summary of the Tree MDP Example



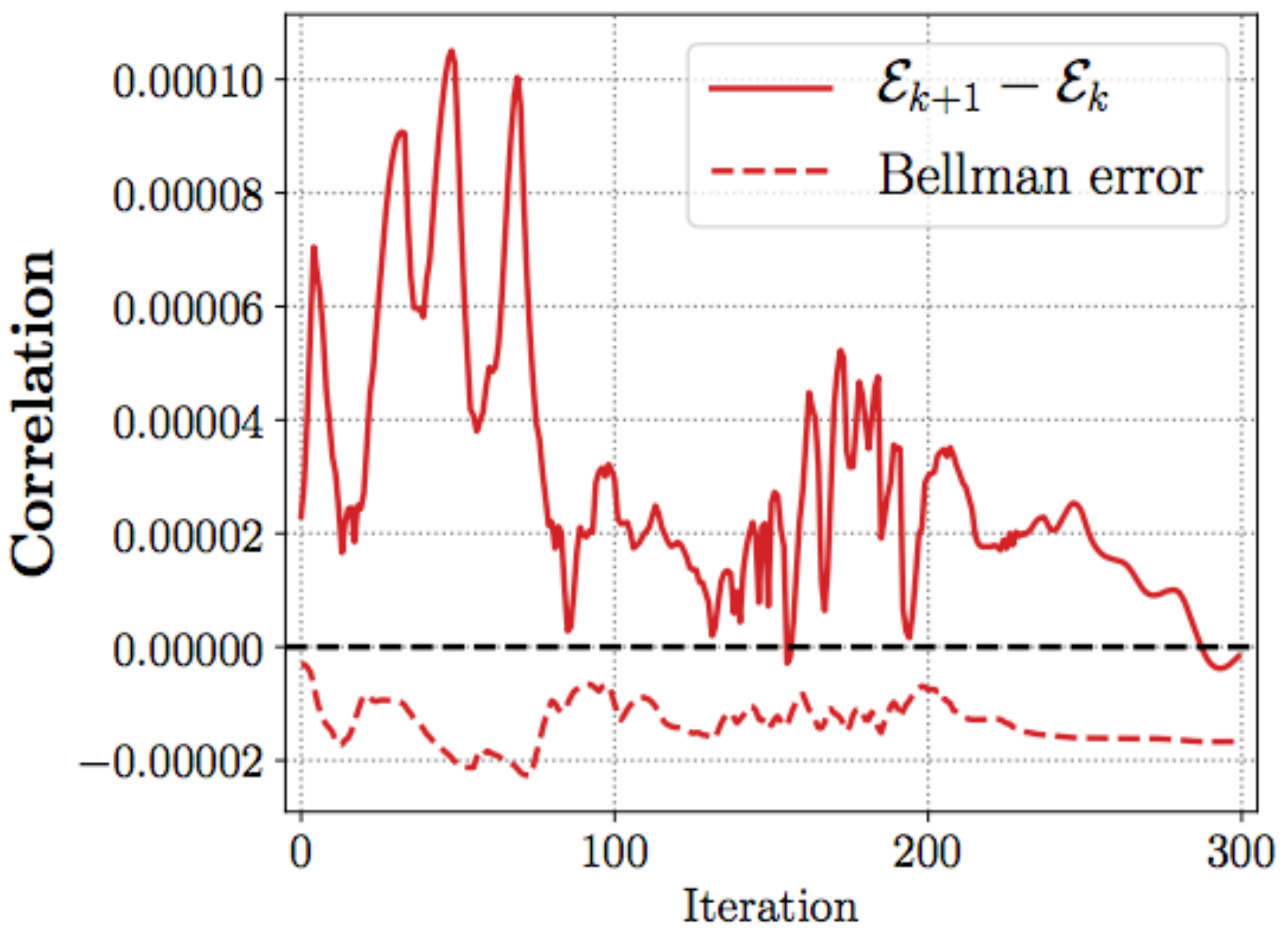
On-policy distributions may not correct errorful targets due to aliasing and skewness

Incorrect target value induces error at other frequent states via Bellman backups

Training under the resulting distribution may further inhibit error correction at leaf nodes

Kumar, Gupta, Levine. DisCor: Corrective Feedback in Reinforcement Learning via Distribution Correction. NeurIPS 2020.

Q-Learning with On-Policy Data Collection



$$d^{\pi_k}(s, a)$$

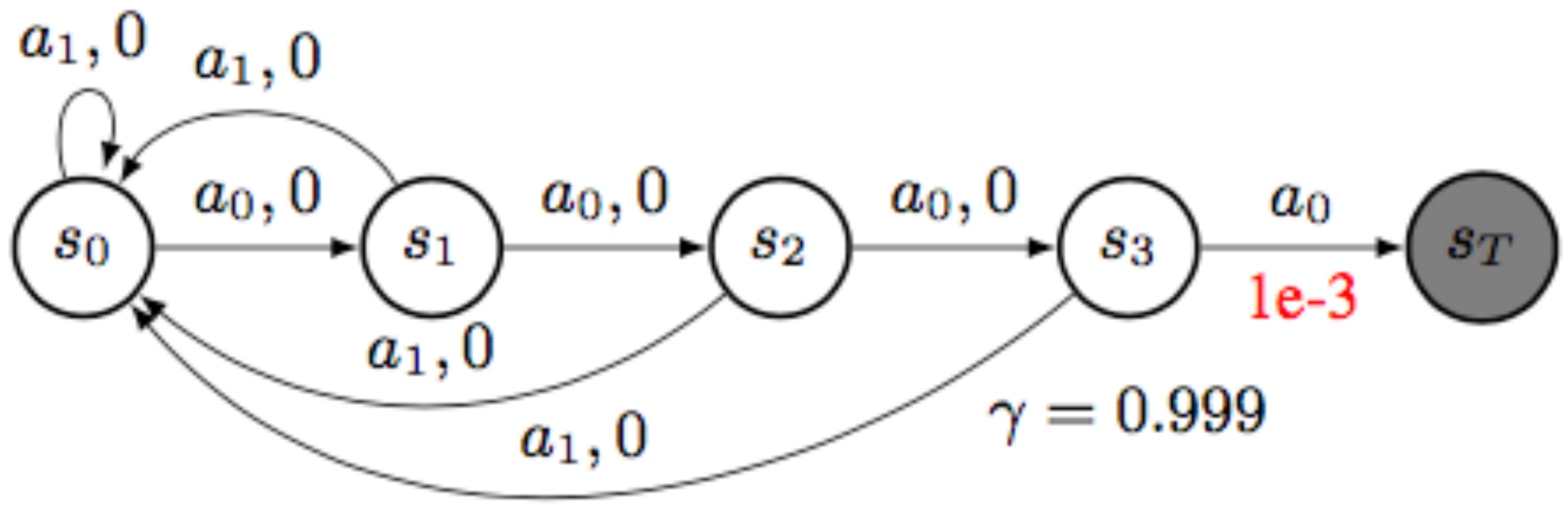
$$\mathcal{E}_k = |Q_k - Q^*|$$

Policy visitation corresponds to reduced Bellman error, but overall error may increase!

$$Q(s, a) = [w_1, w_2]^T \phi(s, a)$$

$$\phi(\cdot, a_0) = [1, 1]$$

$$\phi(\cdot, a_1) = [1, 1.001]$$



$$[w_1, w_2]_{\text{init}} = [0, 1e-4]$$

Check that overall error increases!

Kumar, Gupta, Levine. DisCor: Corrective Feedback in Reinforcement Learning via Distribution Correction. NeurIPS 2020.

What does this tell us?

- While on-policy data collection is sufficient for “error correction”/ convergence in the absence of function approximation, function approximation can make it ineffective in error correction…

- We saw that more gradient updates under such a distribution lead to poor features (due to the implicit under-parameterization phenomenon), which can potentially lead to poor solutions after that….

- We saw that entropic distributions are better — but we have no control over what comes in the buffer, unless we actually change the exploration strategy, so can we do better?

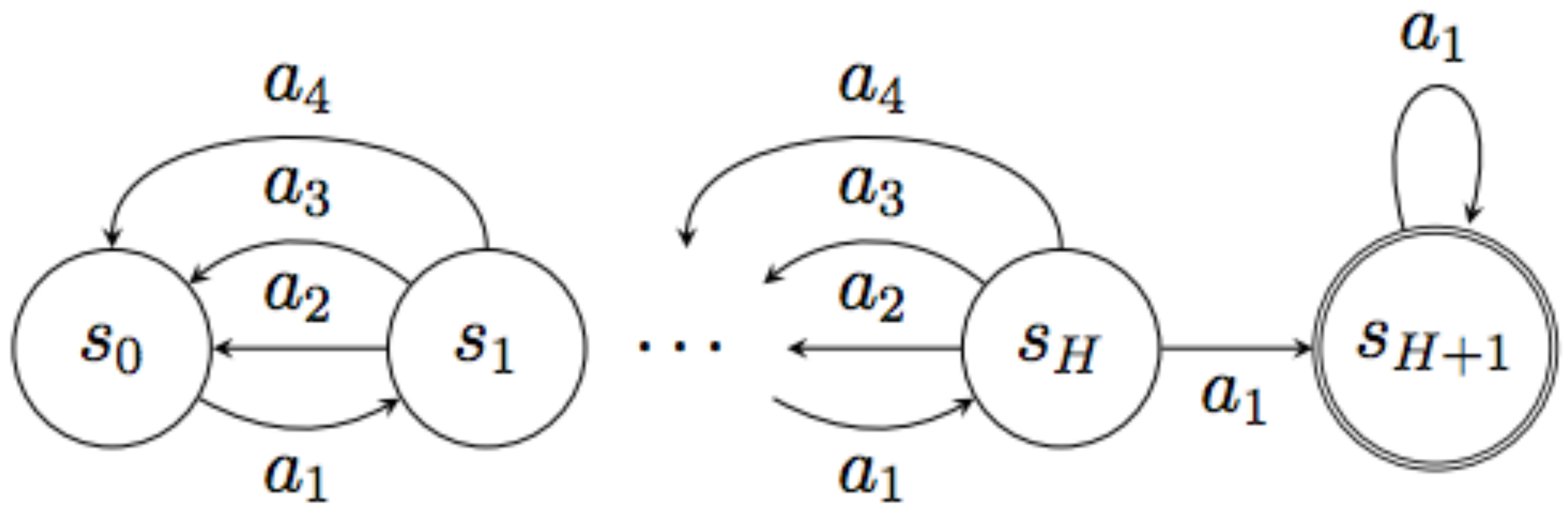
Part 2: Policy Gradient Algorithms

Initial State Distribution in Policy Gradients

Policy gradients maximize expected value at the initial state

$$\max_{\pi} J(\pi) = V^{\pi}(s_0)$$

$$\nabla_\theta V^{\pi_\theta}(s_0) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_{s_0}^{\pi_\theta}} \mathbb{E}_{a \sim \pi_\theta(\cdot|s)} \big[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)\big].$$



Policy gradient can be nearly 0, leading to poor solutions!

Ignore γ^t? Reward shaping? re-weighting?

Proposition 4.1 (Vanishing gradients at suboptimal parameters). Consider the chain MDP of Figure 2, with $H+2$ states, $\gamma = H/(H+1)$, and with the direct policy parameterization (with $3|S|$ ... ise) ... $H/4$, where $\nabla_\theta^k V^{\pi_\theta}(s_0)$ is a tensor of the k_{th} order derivatives of $V^{\pi_\theta}(s_0)$ and the norm is the operator norm of the tensor.[4] Furthermore, $V^\star(s_0) - V^{\pi_\theta}(s_0) \geq (H+1)/8 - (H+1)^2/3^H$.

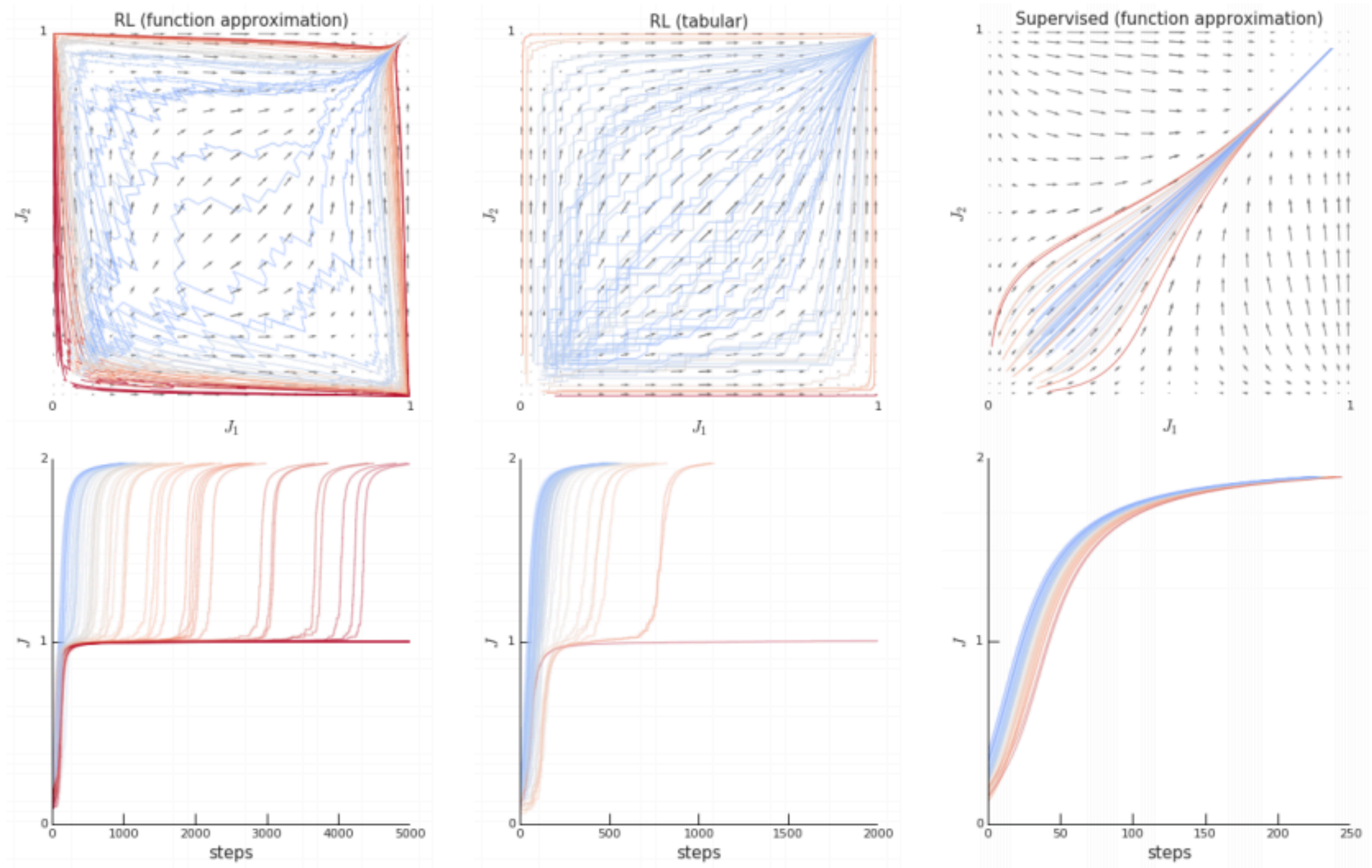
Poor solutions are sort of "equally poor" and the depth of the chain makes it hard to find any gradient of improvement

Agarwal, Kakade, Lee, Mahajan. On the Theory of Policy Gradient Methods. 2019

Policy Gradient Plateaus: What and Why?

Policy gradient + function approximation + on-policy data

$J_1 + J_2$



Might end up optimizing one component of the objective more than others

If you hit a saddle point of the expected return function (corners), then stays there

Initialization, etc become important now!

$$\pi_\theta(a|s) = \frac{\exp\{\theta(s,a)\}}{\sum_{a'} \exp\{\theta(s,a')\}}.$$

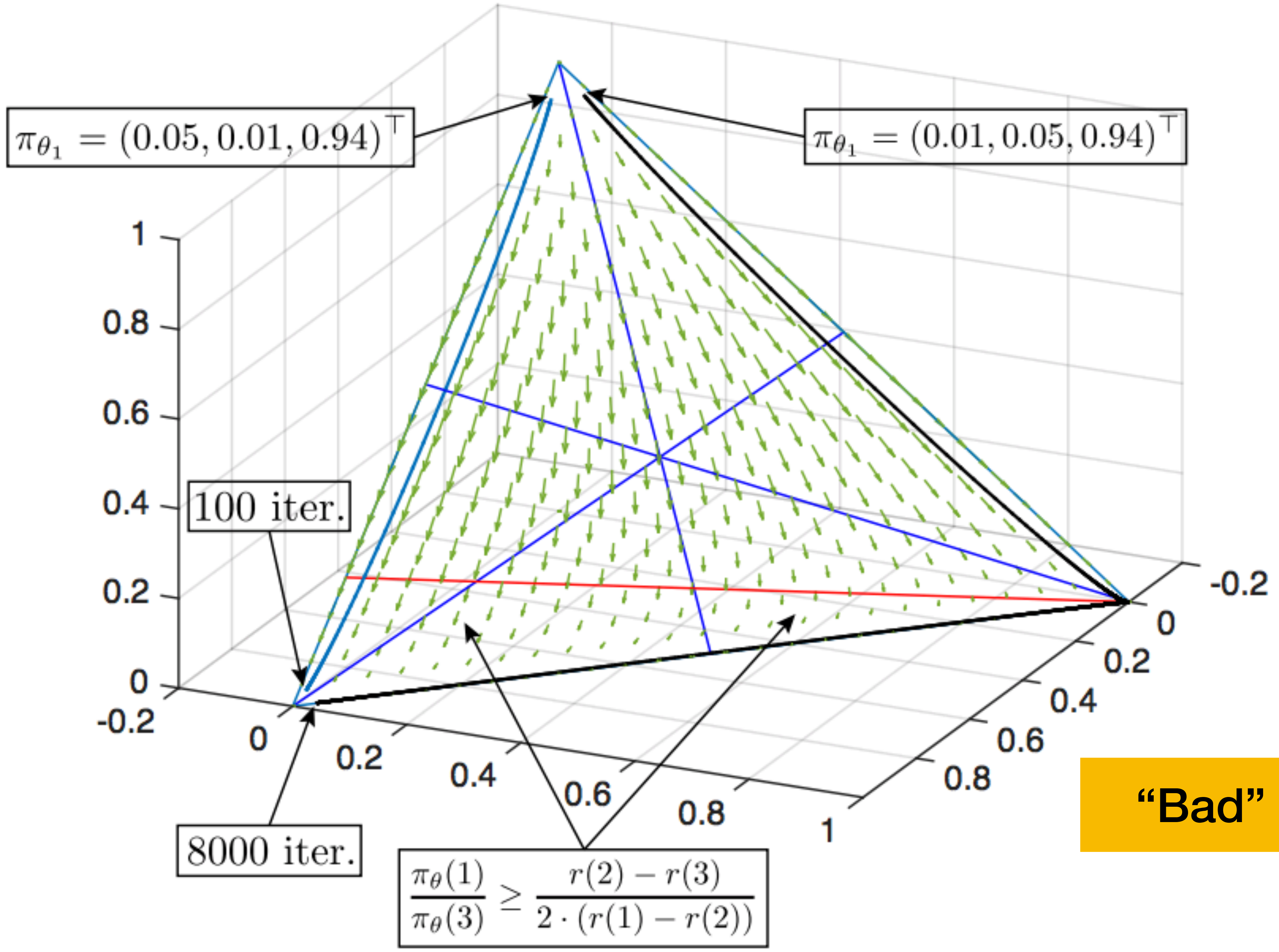
Also affected by attraction to suboptimal solutions during training!

Schaul, Borsa, Modayil, Pascanu. Ray interference: A source of plateaus in deep RL. 2019

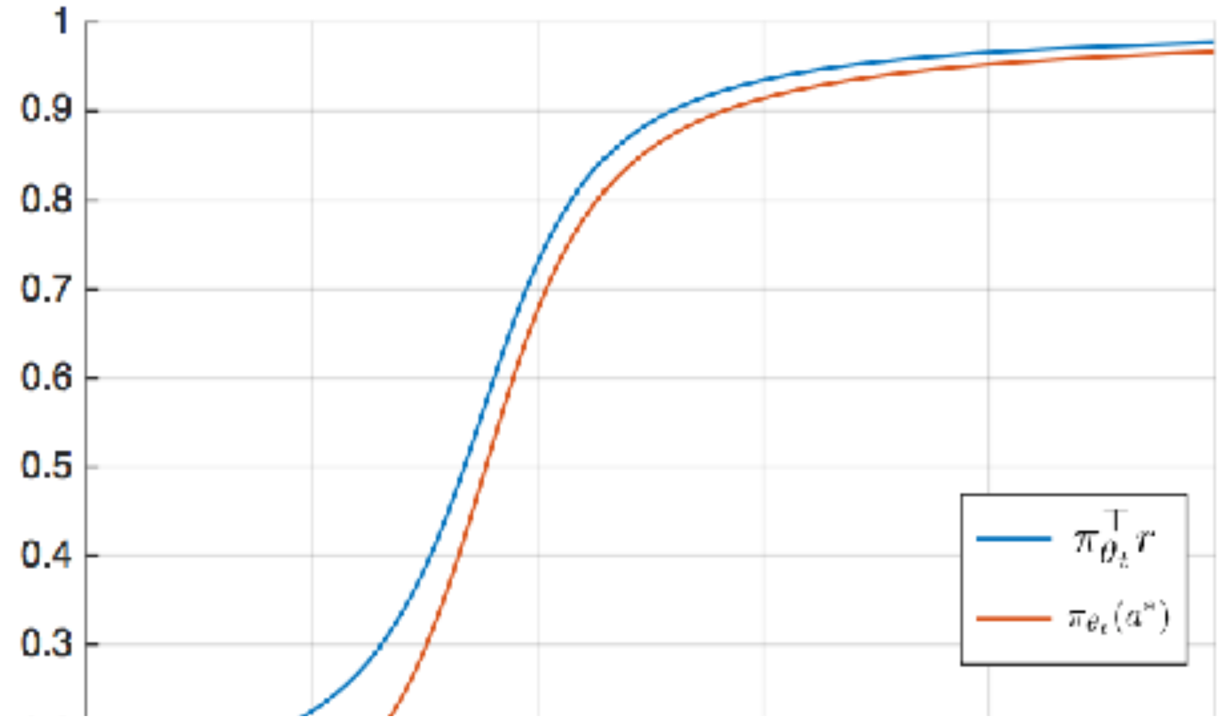
Importance of Initialization and Rewards

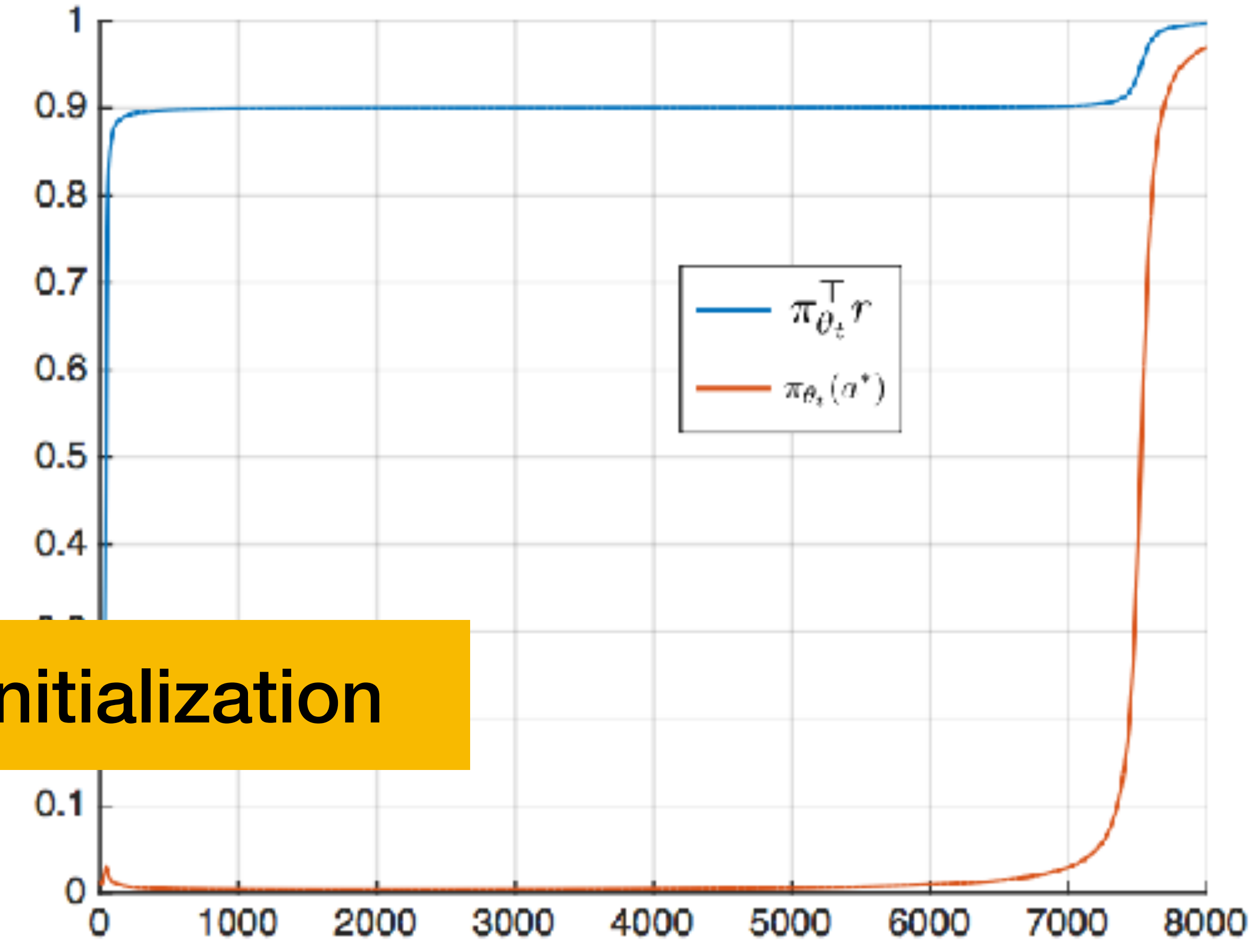
Reward values affect policy gradient methods a lot!

Consider a 3-armed bandit (1-step RL) problem with arm 1 being the optimal arm.



“Good” initialization





“Bad” initialization

Initialization matters, reward values matter. Also matter in offline settings.

Mei, Xiao, Szepesvari, Schuurmans. On the Global Convergence of Softmax Policy Gradient Methods. ICML 2020.

Summary and Takeaways

- Overfitting in RL consists of more than just sampling error in standard supervised learning — we discussed how the update in Q-learning leads to poor solutions.

- Data-distributions matter a lot for RL problems: for both Q-learning algorithms and policy-gradient algorithms, only started understanding the surface in this domain

- Iterated training and changing objectives can be heavily affected by initialization, coverage, function approximation, etc in both Q-learning and policy gradient methods

Several open questions along these lines, have the potential to lead to stable and efficient algorithms