Better Actor-Critic Algorithms for Reinforcement Learning

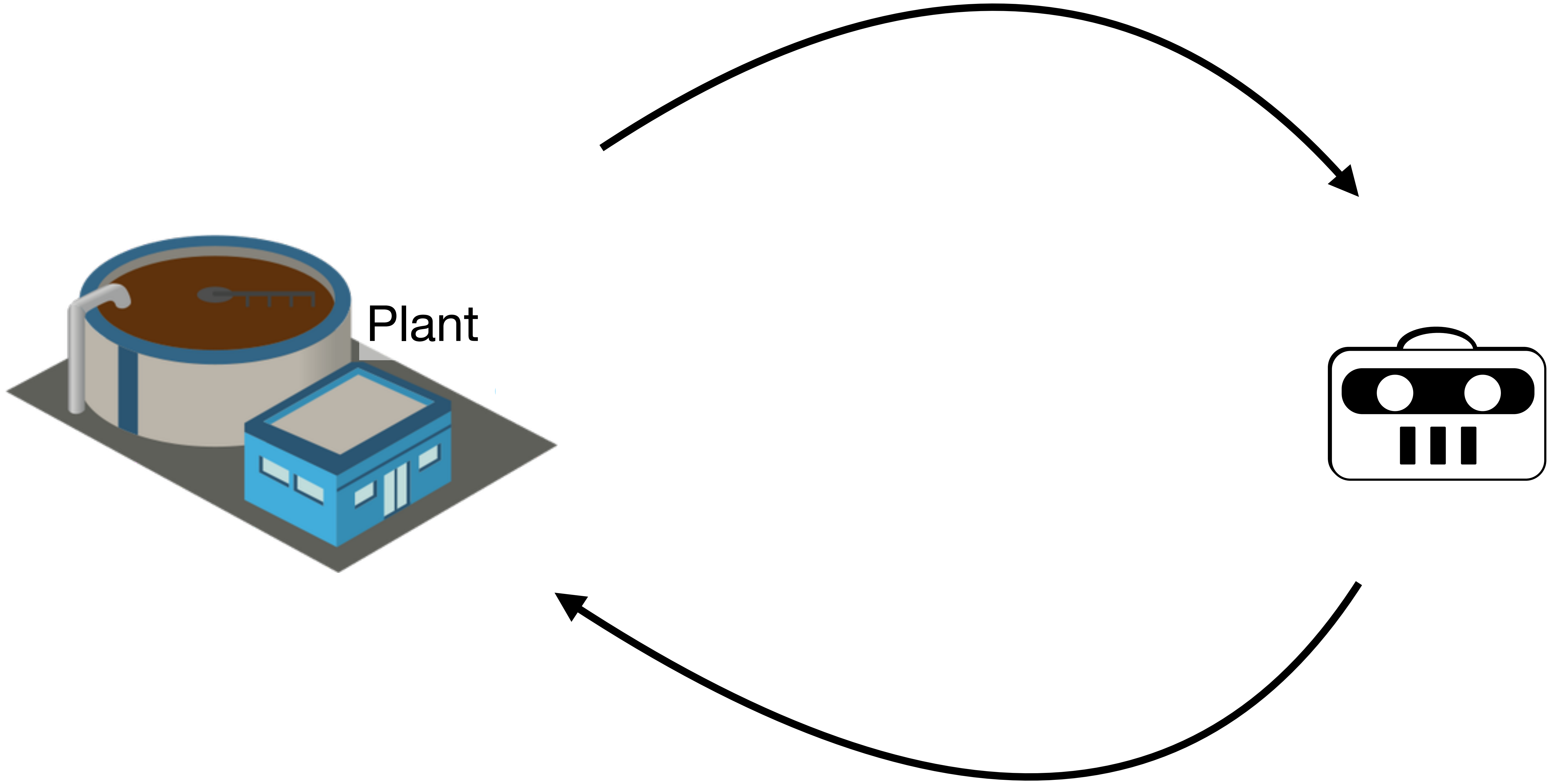
Martha White
Associate Professor
University of Alberta
Canada CIFAR AI Chair and Fellow of Amii
Canada Research Chair in Reinforcement Learning

My Goal Today

- Motivate why we need **better actor-critics**
- Discuss how we can see actor-critic methods as approximate policy iteration
- Highlight three key choices underlying many existing actor-critic methods
    - and a little bit about what the theory says about them
- Describe our algorithm, called Greedy Actor-Critic
    - focused on improving one of these three choices

Online Reinforcement Learning Setting

- Imagine an agent that is continually optimizing operation of a drinking water treatment plant




Mixing speed
Backwash aeration
Backwash onset
Backwash duration
Backwash cycle
Chemical dosing
*image from partner ISL Engineering

Need online, single-stream algorithms that can run for a long time

Bold claim: Our deep RL algorithms to learn policies are bad

Bold claim: Our deep RL algorithms to learn policies are bad

They are notoriously finicky with lots of hyperparameters

We layer on more tricks, because they aren’t working well

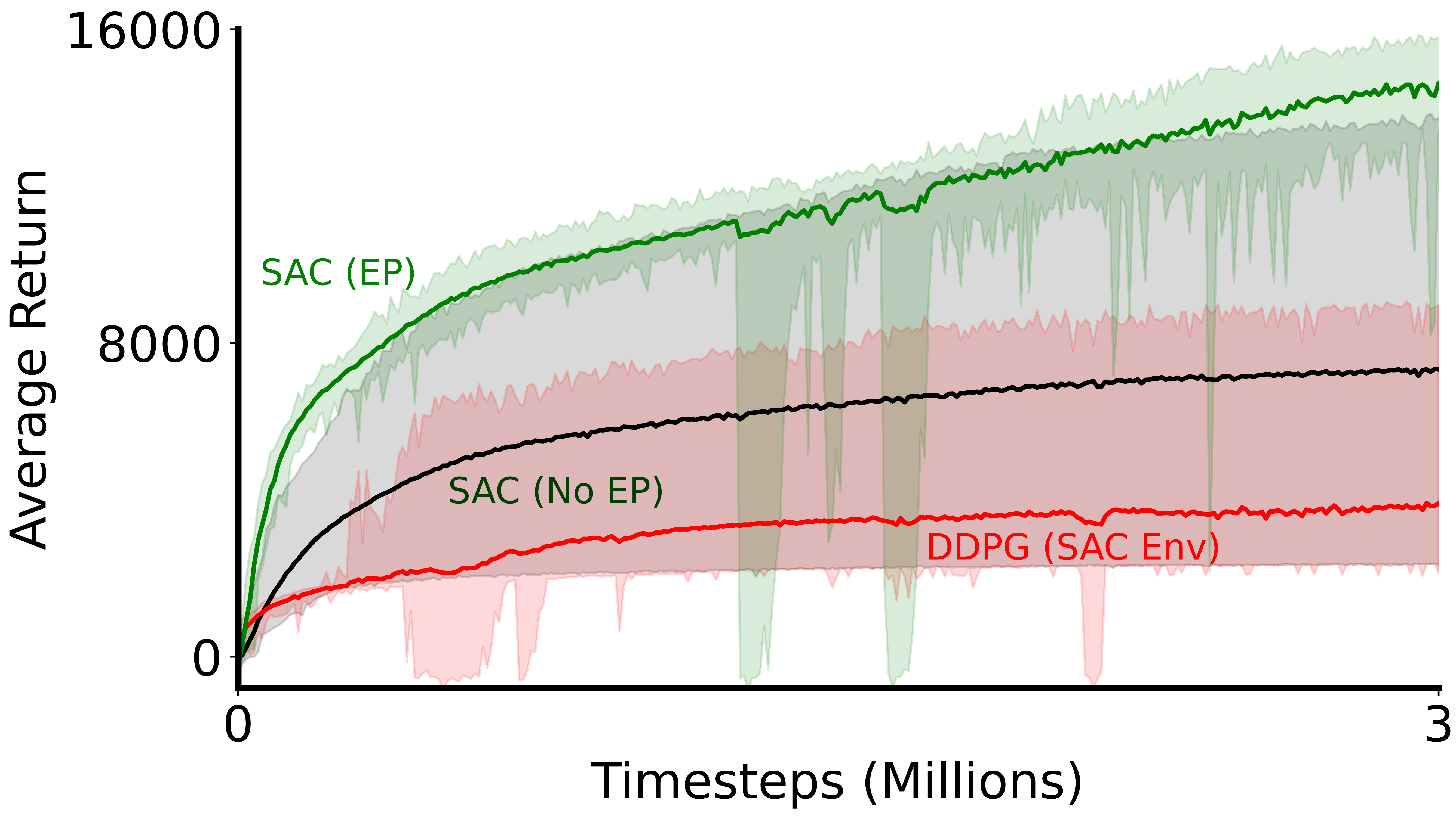
Case Study: Recreating a Result for Soft-Actor Critic (SAC)

- Soft-Actor Critic is a commonly-used algorithm
- For our paper on how to do better experiments in RL, we included a case study recreating SAC's results on an environment called Half-Cheetah

* See our paper: "Empirical Design in Reinforcement Learning", Patterson et al., 2024

Significant difference from one implementation detail

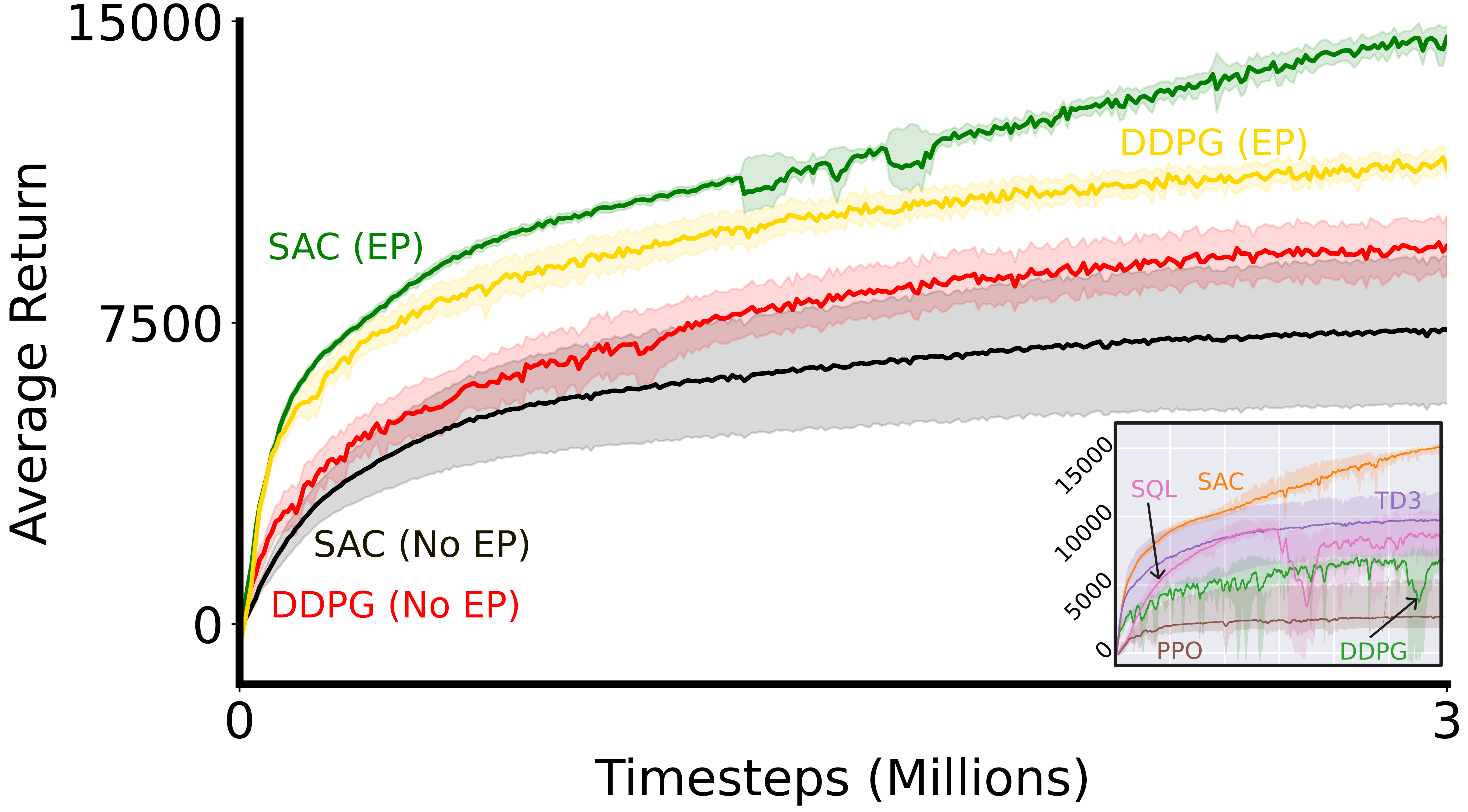
- Black line is SAC implemented using only details in the paper
- DDPG (Deep Deterministic Policy Gradient) is a baseline in their work
- **A key detail found in their code was to add an exploration phase**, SAC (EP)



* See our paper: "Empirical Design in Reinforcement Learning", Patterson et al., 2024

Adding implementation-level improvements to DDPG

- DDPG becomes competitive when
  (a) reconsidering the noise process for exploration
  (b) adding exploration phase
- Inset plot is from their work



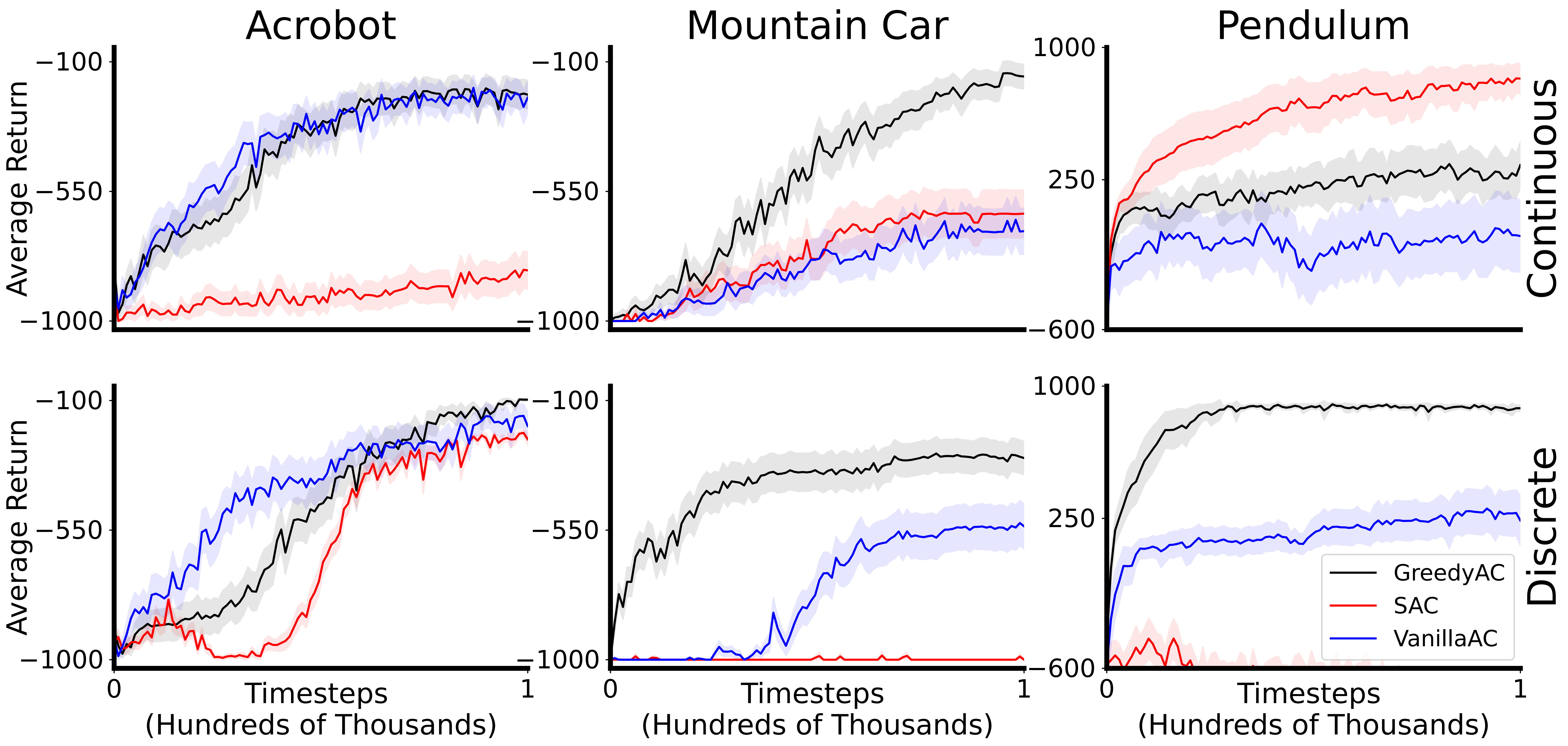
* See our paper: "Empirical Design in Reinforcement Learning", Patterson et al., 2024

But now we have a working understanding of SAC,
and can use it elsewhere

SAC is failing on classic control environments

- Many in RL would say these environments are too simple

*entropy, critic & actor stepsize tuned **across** environments



Our current goal is to remove (subtract not add)

And get a more minimal AC algorithm, inspired by theory

To really understand Actor-Critic and the theory behind it

let’s talk about Actor-Critic as Approximate Policy Iteration

Refresher on Policy Iteration

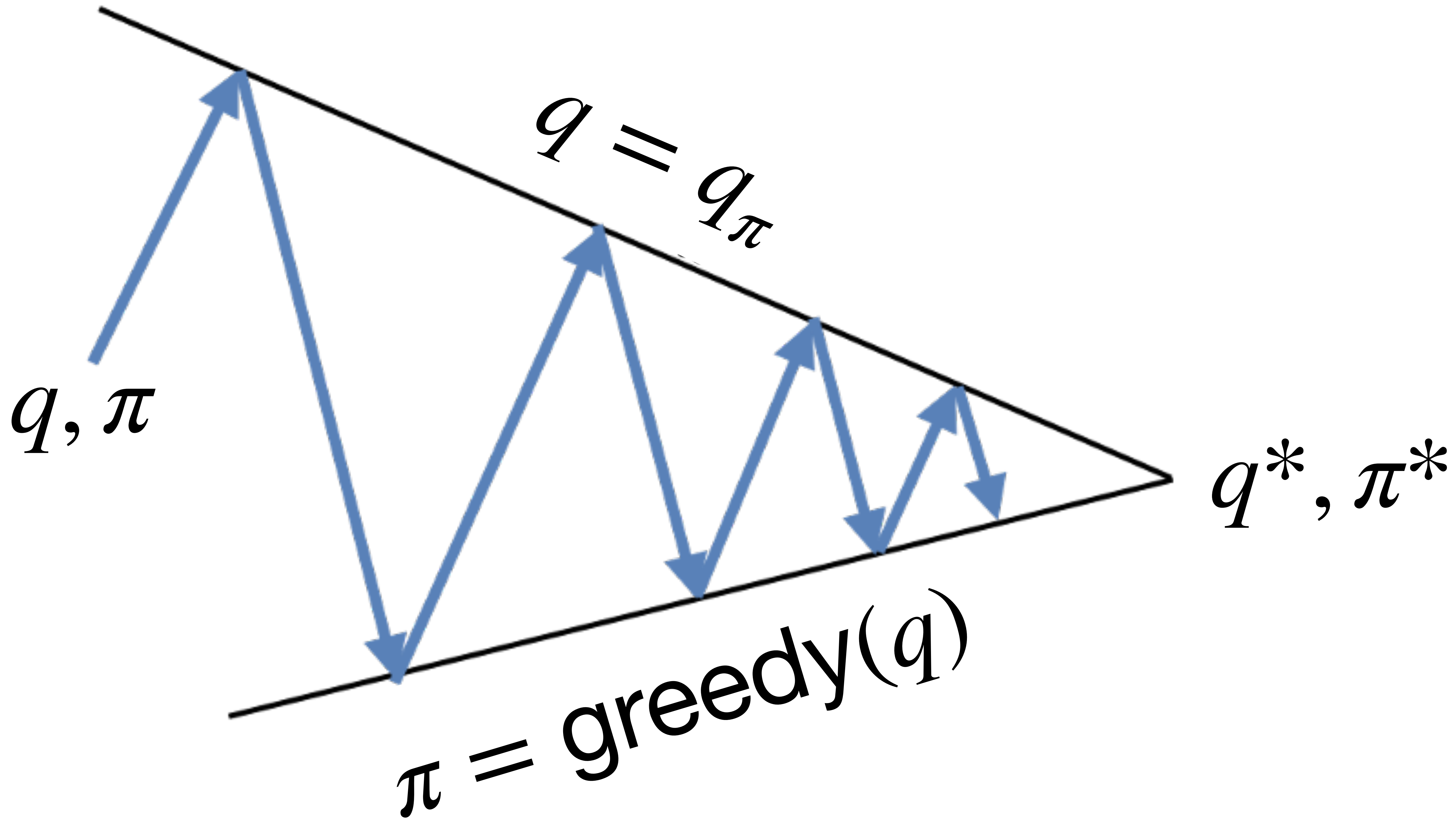
- Policy iteration is built on a foundational result:
  the policy improvement theorem

The Policy Improvement Theorem

- For the current policy π and action-values q_π
- if we get the new policy π' by making it greedy in q_π
  - e.g., $\pi'(s) = \arg\max_{a \in \mathcal{A}} q_\pi(s, a)$
- then π' is guaranteed to be at least as good as π

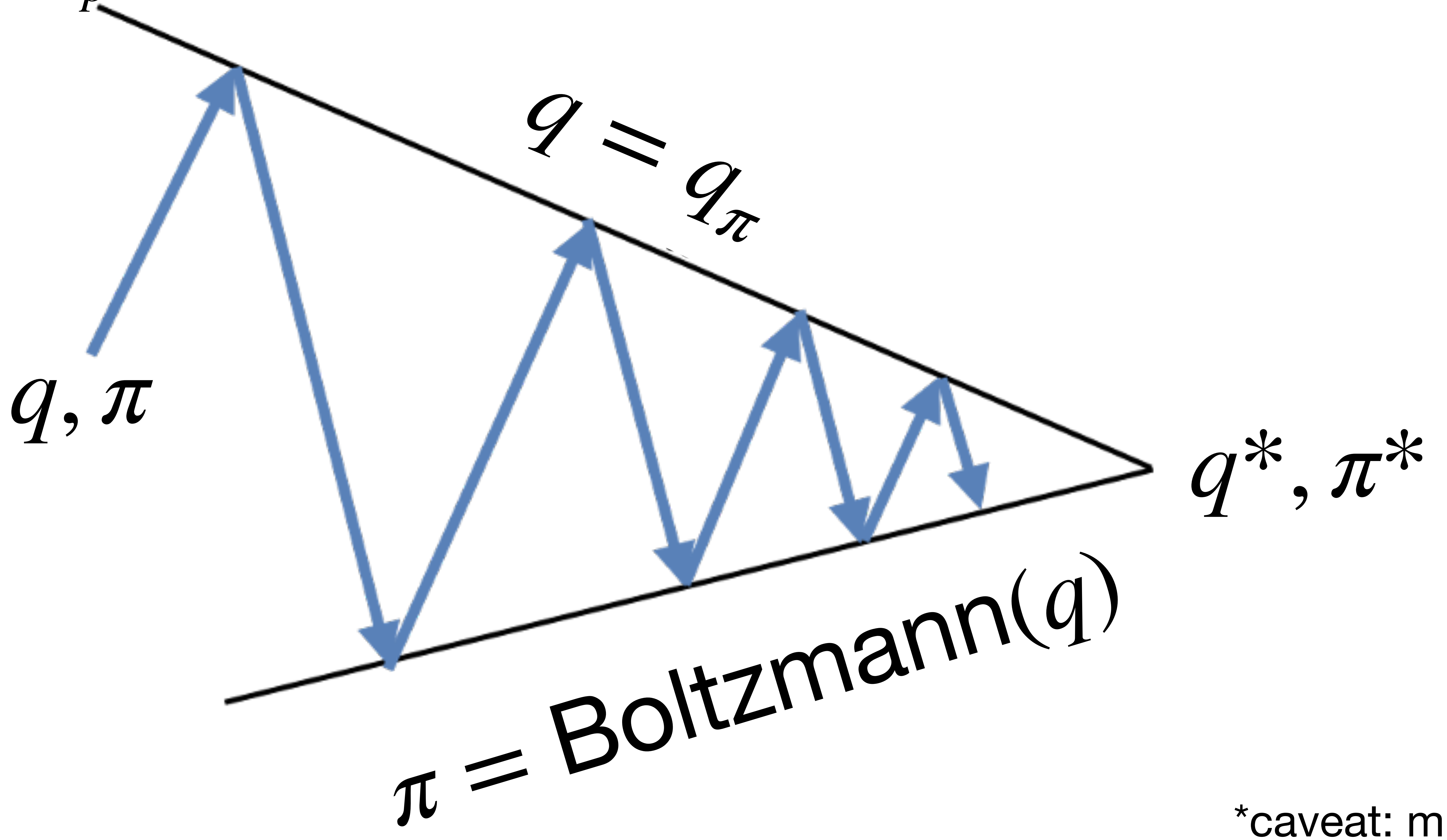
Policy Iteration

- For the current policy π and action-values q_π
- Get new policy π' by making it greedy in q_π, then obtain $q_{\pi'}$ and repeat



Entropy-Regularized Greedification

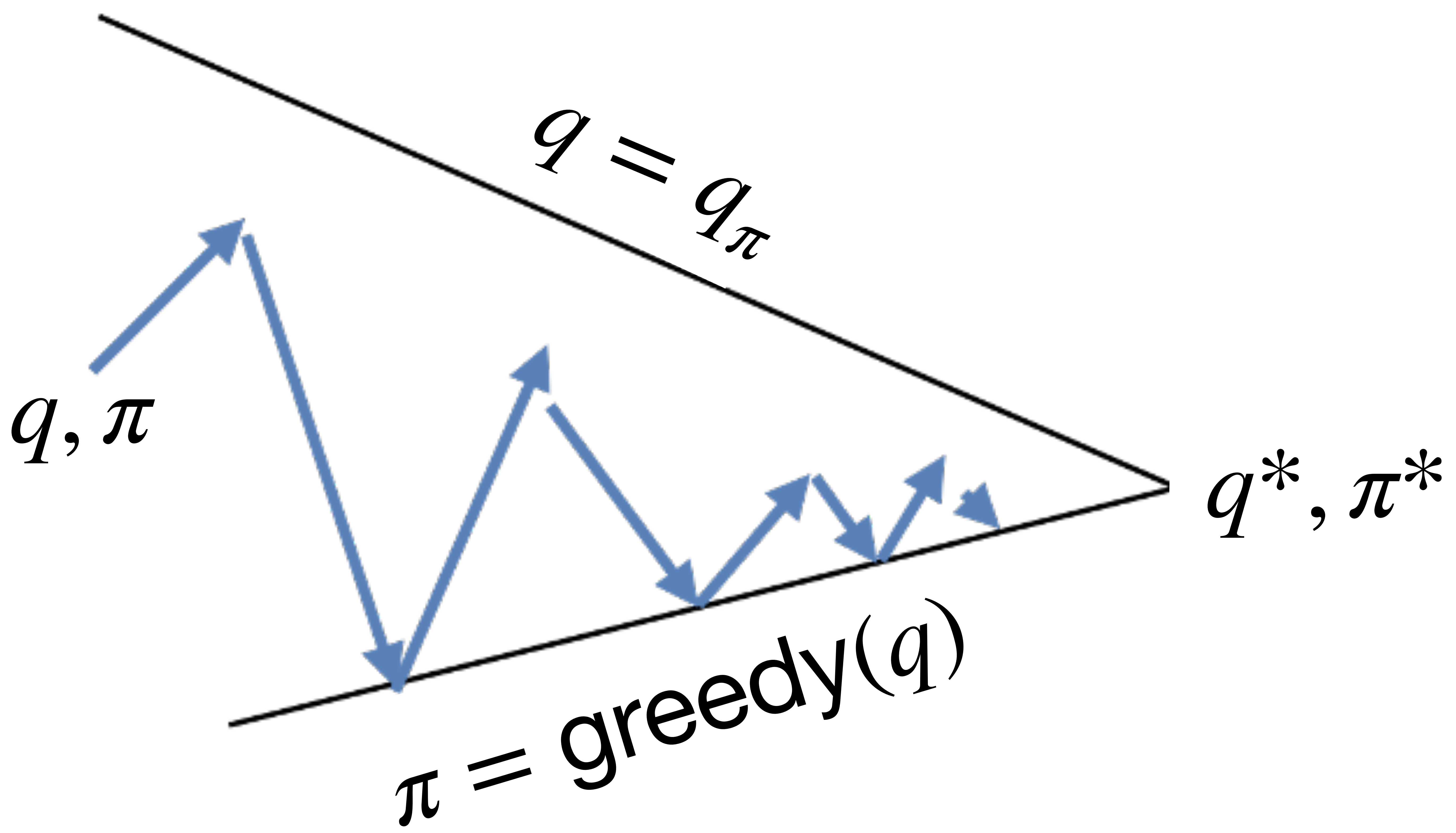
- Can also get new policy π' by making it soft-greedy in q_π,
- $\pi_{\text{ent}}(\cdot \mid s) = \arg\max_{p} \mathbb{E}_{a\sim p}[q(s,a)] + \tau\mathscr{H}(p) = \text{Boltzmann}(q(s,\cdot)/\tau) \propto \exp(q(s,\cdot)/\tau)$



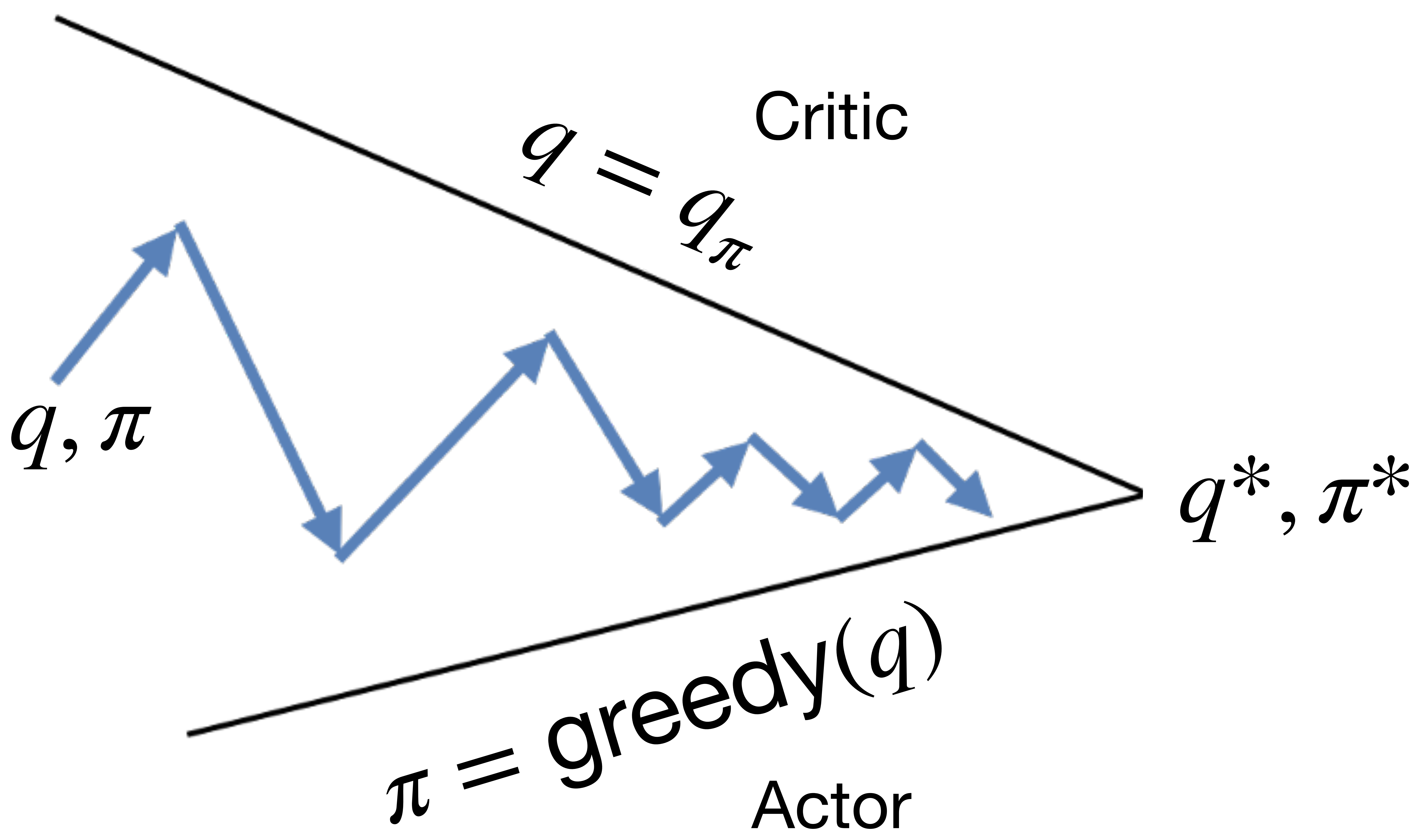
*caveat: must use soft action-values

In reality, we do **Approximate Policy Iteration**

Approximate Policy Evaluation



Approximate Greedification and Evaluation



The class of **Actor-Critic** algorithms can be seen as doing API

A Representative Actor-Critic Algorithm

- The agent interacts with the environment, taking actions $a \sim \pi_\theta(\,\cdot \mid s)$
- It stores all that data in a replay buffer, to do mini-batch updates each step
- Buffer $= \{(s_0, a_0, r_1, s_1), (s_1, a_1, r_2, s_2), (s_2, a_2, r_3, s_3), \ldots, (s_{t-1}, a_{t-1}, r_t, s_t)\}$

An Actor-Critic Update with Replay

- Sample (s, a, r, s') from the replay buffer (or sample a mini-batch)
- **Update critic** q_w using Sarsa for prediction on (s, a, r, s')
  - Update moves q_w closer to q_{π_θ} (approximate policy evaluation)

An Actor-Critic Update with Replay

- Sample (s, a, r, s') from the replay buffer (or sample a mini-batch)
- Update critic q_w on (s, a, r, s')
- **Update actor** π_θ using the log-likelihood update

$$\tilde{a} \sim \pi_\theta(\,\cdot \mid s)$$
$$\theta \leftarrow \theta + \eta q_w(s, \tilde{a}) \nabla \ln \pi_\theta(\tilde{a} \mid s)$$

Update increases $\mathbb{E}_{a \sim \pi_\theta(\cdot \mid s)}[q(s, a)]$, likelihood of actions with high value under q_w (greedifies)

Entropy-regularized Actor-Critic Update

- Sample (s, a, r, s') from the replay buffer (or sample a mini-batch)
- Update critic q_w on (s, a, r, s')
- **Update actor** π_θ using the log-likelihood update

$$\tilde{a} \sim \pi_\theta(\cdot \mid s)$$
$$\theta \leftarrow \theta + \eta q_w(s, \tilde{a}) \nabla \ln \pi_\theta(\tilde{a} \mid s) + \eta \nabla \mathscr{H}(\pi_\theta(\cdot \mid s))$$

Update increases $\mathbb{E}_{a \sim \pi_\theta(\cdot|s)}[q(s, a)]$, likelihood of actions with high value under q_w while ensuring entropy stays higher (greedifies)

An Actor-Critic Update with Replay

- **Sample** (s, a, r, s') from the replay buffer (or sample a mini-batch)
- **Update critic** q_w using Sarsa on (s, a, r, s')
- **Update actor** π_θ using the log-likelihood update

$$\tilde{a} \sim \pi_\theta(\cdot \mid s)$$
$$\theta \leftarrow \theta + \eta q_w(s, \tilde{a}) \nabla \ln \pi_\theta(\tilde{a} \mid s)$$

Update increases $\mathbb{E}_{a \sim \pi_\theta(\cdot|s)}[q(s, a)]$, likelihood of actions with high value under q_w (greedifies)

Three Key Choices for Many Actor-Critic Algorithms

For a given state s

1. How should we update the **critic** q? (do approximate policy evaluation)

2. How should we update the **actor** π? (do approximate greedification)

Three Key Choices for Many Actor-Critic Algorithms

For a given state s

1. How should we update the **critic** q? (do approximate policy evaluation)

2. How should we update the **actor** π? (do approximate greedification)

3. How much importance (weight) do we put on each state?

Three Key Choices for Many Actor-Critic Algorithms

For a given state s

1. How should we update the critic q? (do approximate policy evaluation)

2. How should we update the actor π? (do approximate greedification)

3. How much importance (weight) do we put on each state?*

- certain choices can cause very suboptimal behavior
- we solved an open problem (proved an off-policy policy gradient theorem) and used this theoretical result to get a sound algorithm

* See our recent journal paper: “Actor Critic with Emphatic Weightings” Graves et al., JMLR, 2023

Outcome of the theorem

- Objective is $J(\theta) = \mathbb{E}_{s \sim \mu, a \sim \pi_\theta(\cdot|s)}\left[q_{\pi_\theta}(s, a)\right]$
- where μ is the state distribution (e.g., distribution over states in data)
- Underlying update used by many actor-critic methods

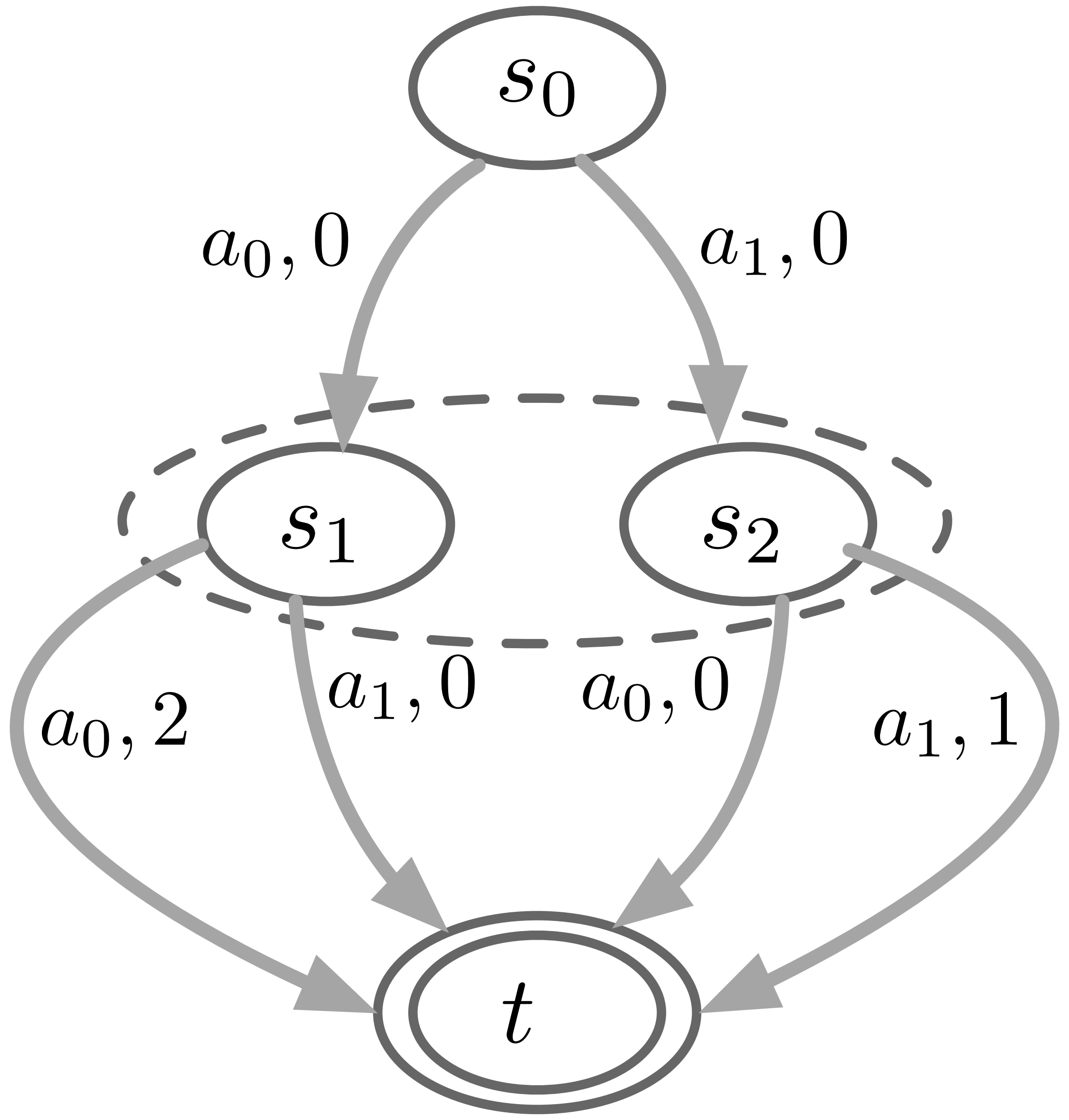
$$\nabla J(\theta) = \mathbb{E}_{s \sim \mu, a \sim \pi_\theta(\cdot|s)}\left[q_{\pi_\theta}(s, a) \nabla \ln \pi_\theta(a \mid s)\right]$$

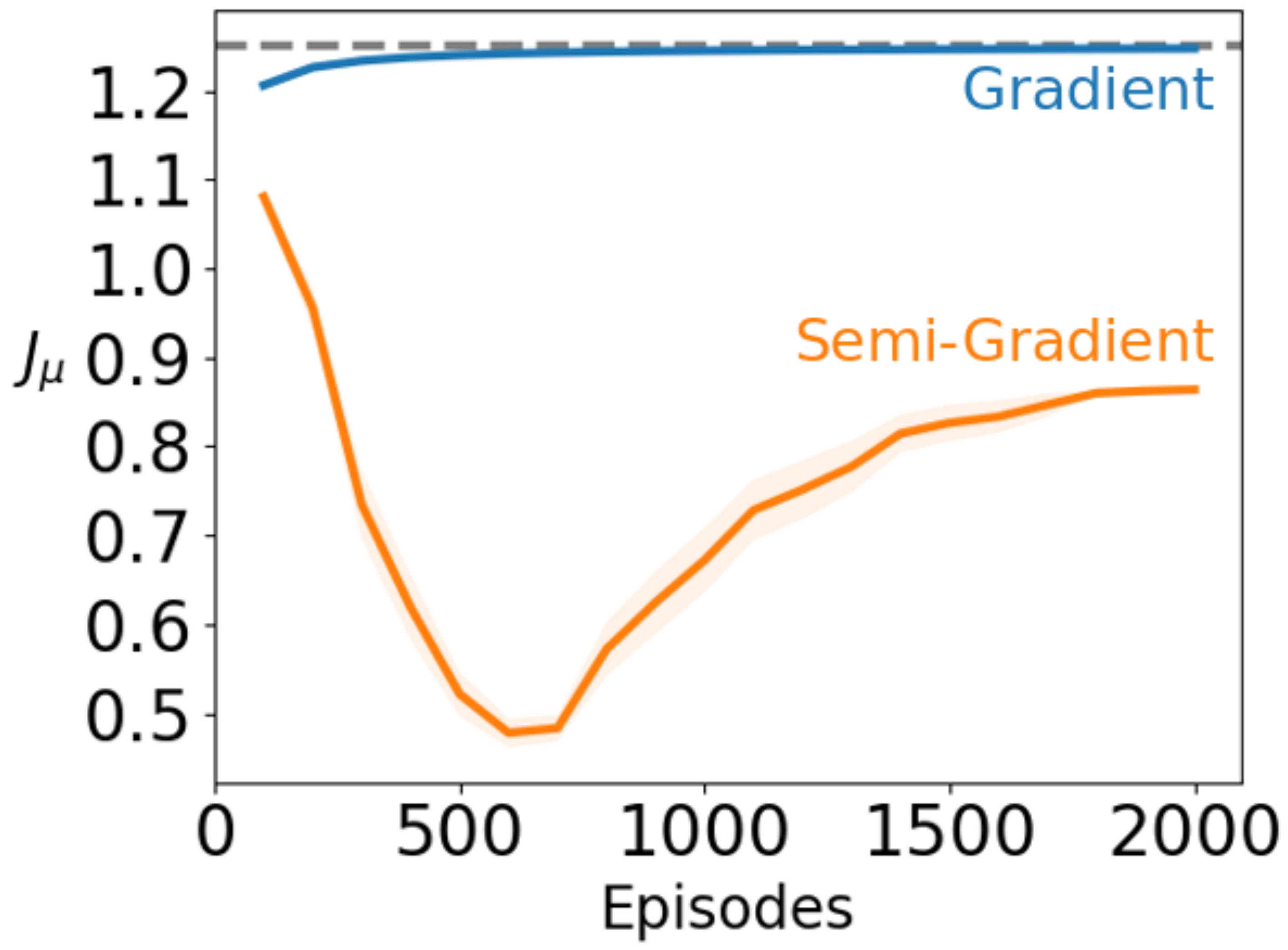
- Correct gradient requires a reweighting, with m the emphatic weight

$$\nabla J(\theta) = \mathbb{E}_{s \sim m, a \sim \pi_\theta(\cdot|s)}\left[q_{\pi_\theta}(s, a) \nabla \ln \pi_\theta(a \mid s)\right]$$

* See our journal paper: “Actor Critic with Emphatic Weightings” Graves et al., JMLR, 2023

Suboptimal policy under standard off-policy AC





- **Semi-gradient** (standard off-policy AC) updates with $s \sim$ stationary distribution under behavior policy
- **Gradient** reweights updates with emphatic weightings

* See our journal paper: “Actor Critic with Emphatic Weightings” Graves et al., JMLR, 2023

Three Key Choices for Many Actor-Critic Algorithms

For a given state s

1. How should we update the critic q? (do approximate policy evaluation)

2. How should we update the actor π? (do approximate greedification)

3. How much importance (weight) do we put on each state?*

- certain choices can cause very suboptimal behavior
- but state weighting only impacts how we trade-off accuracy under limited function approximation (e.g., no suboptimality in tabular setting)

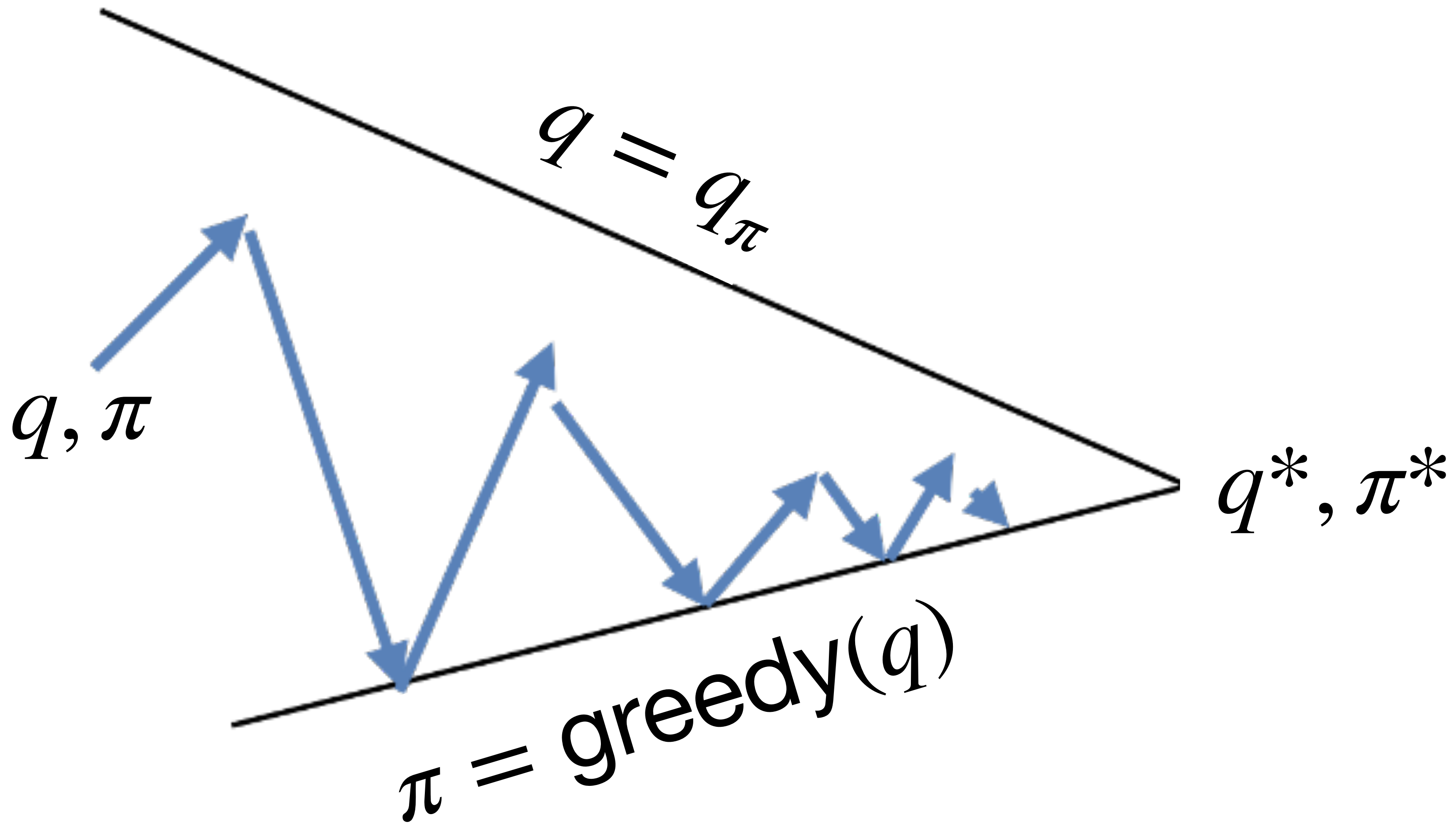
* See our journal paper: “Actor Critic with Emphatic Weightings” Graves et al., JMLR, 2023

Three Key Choices for Many Actor-Critic Algorithms

For a given state s

1. How should we update the critic q? (do approximate policy evaluation)

- Recent theory accounts for some error in q, with exact greedification* when using a KL to the previous policy, i.e., mirror descent update



* See nice papers on MD-MPI (Vieillard et al, 2020), Politex (Abbasi-Yadkori et al., 2019)

Three Key Choices for Many Actor-Critic Algorithms

For a given state s

1. How should we update the critic q? (do approximate policy evaluation)

2. **How should we update the actor π? (do approximate greedification)**

    - lots of theory for unbiased/exact policy evaluation (policy gradient)

    - **but** what about approximate policy evaluation and greedification?

Why can't we always do exact greedification?

Reason 1

- For discrete actions, can always exactly use the (soft) greedy policy
  - e.g., exactly set $\pi(a \mid s) = \pi_{\text{ent}}(a \mid s) = \dfrac{\exp(q(s,a)/\tau)}{\sum_b \exp(q(s,b)/\tau)}$

Why can't we always do exact greedification?

Reason 1

- For discrete actions, can always exactly use the (soft) greedy policy
  - e.g., exactly set $\pi(a \mid s) = \pi_{\text{ent}}(a \mid s) = \dfrac{\exp(q(s,a)/\tau)}{\sum_b \exp(q(s,b)/\tau)}$
- But! For continuous actions, sampling Boltzmann$(q(s, \cdot\,))$ is expensive

Why can't we always do exact greedification?

Reason 2

- Even for discrete actions, it is common to add a KL divergence (with weight λ) to the previous policy π_{t-1}
  - want $\pi_t = \pi_{\text{kl}}$ where $\pi_{\text{kl}}(a \mid s) \propto \pi_{t-1}(a \mid s) \exp(q(s, a)/\lambda)$

Why can't we always do exact greedification?

Reason 2

- Common to add a KL divergence (with weight λ) to the previous policy π_{t-1}
    - want $\pi_t = \pi_{\text{kl}}$ where $\pi_{\text{kl}}(a \mid s) \propto \pi_{t-1}(a \mid s) \exp(q(s,a)/\lambda)$
- Unrolling, we get $\pi_{\text{kl}}(a \mid s) \propto \exp\left(\frac{1}{\lambda}\sum_{j=0}^{t} q_j(s,a)\right)$
- Getting this policy requires averaging all previous critics q_j (!!)
    - even for discrete actions

Approximate greedification for Boltzmann

- Move parameterized policy π_θ closer to this desired policy
  - reduce KL divergence between π_θ and π_{ent}
  - $\theta \leftarrow \theta - \alpha \nabla_\theta \text{KL}(\pi_\theta(\cdot \mid s) \mid\mid \pi_{\text{ent}}(\cdot \mid s))$

Note: this gradient actually gives us the same log likelihood update
with entropy regularization

$$-\tau \nabla_\theta \text{KL}(\pi_\theta(\cdot \mid s) \mid\mid \pi_{\text{ent}}(\cdot \mid s)) = \mathbb{E}_{a \sim \pi_\theta(\cdot \mid s)}[\boxed{q(s, a) \nabla \ln \pi_\theta(a \mid s)}] + \tau \nabla \mathscr{H}(\pi_\theta(\cdot \mid s))$$

Many actor-critic methods use an update like this one

Approximate greedification for KL-policy

- Move parameterized policy π_θ closer to this desired policy
  - or reduce KL divergence between π_θ and π_{kl}
  - $\theta \leftarrow \theta - \alpha \nabla_\theta \text{KL}(\pi_\theta(\cdot \mid s) \,||\, \pi_{\text{kl}}(\cdot \mid s))$

An aside: there are two completely different uses for a KL here
Role 1: **KL penalty** to the previous policy to define the target policy π_{kl}
Role 2: **KL loss** for the actor update

$$-\lambda \nabla_\theta \text{KL}(\pi_\theta(\cdot \mid s) \,||\, \pi_{\text{kl}}(\cdot \mid s)) = \mathbb{E}_{a \sim \pi_\theta(\cdot \mid s)}[q(s,a) \nabla \ln \pi_\theta(a \mid s)] + \lambda \nabla \text{KL}(\pi_\theta(\cdot \mid s) \,||\, \pi_{t-1}(\cdot \mid s))$$

Three Key Choices for Many Actor-Critic Algorithms

For a given state s

1. How should we update the critic q? (do approximate policy evaluation)

2. **How should we update the actor π? (do approximate greedification)**

    - improvement guarantee iff KL reduction greater than difference in average critic error under the new and old policy*

    - main point: complicated interaction between critic error and approximation in greedification step

* See Corollary 9 in our journal paper: "Greedification Operators for Policy Optimization: Investigating Forward and Reverse KL Divergences", Chan et al., JMLR, 2022

Brief summary so far

- Actor-critic algorithms do approximate policy iteration
- Most theory about solution quality either for
  - approximate policy evaluation, exact greedification to π_{kl} (MD-MPI, Politex, Munchausen RL, Implicit Q-values)
  - unbiased/exact policy evaluation, approximate greedification (REINFORCE, CPI, NPG, TRPO, SAC theory, MPO theory, AC with emphatic weightings, FMA-PG)
- When both steps are approximate, need to be more careful about interactions between errors
  - and maybe work extra hard to do each step well

There is so much to do, what shall we tackle?

One direction is to reconsider this reverse KL underlying many AC algorithms

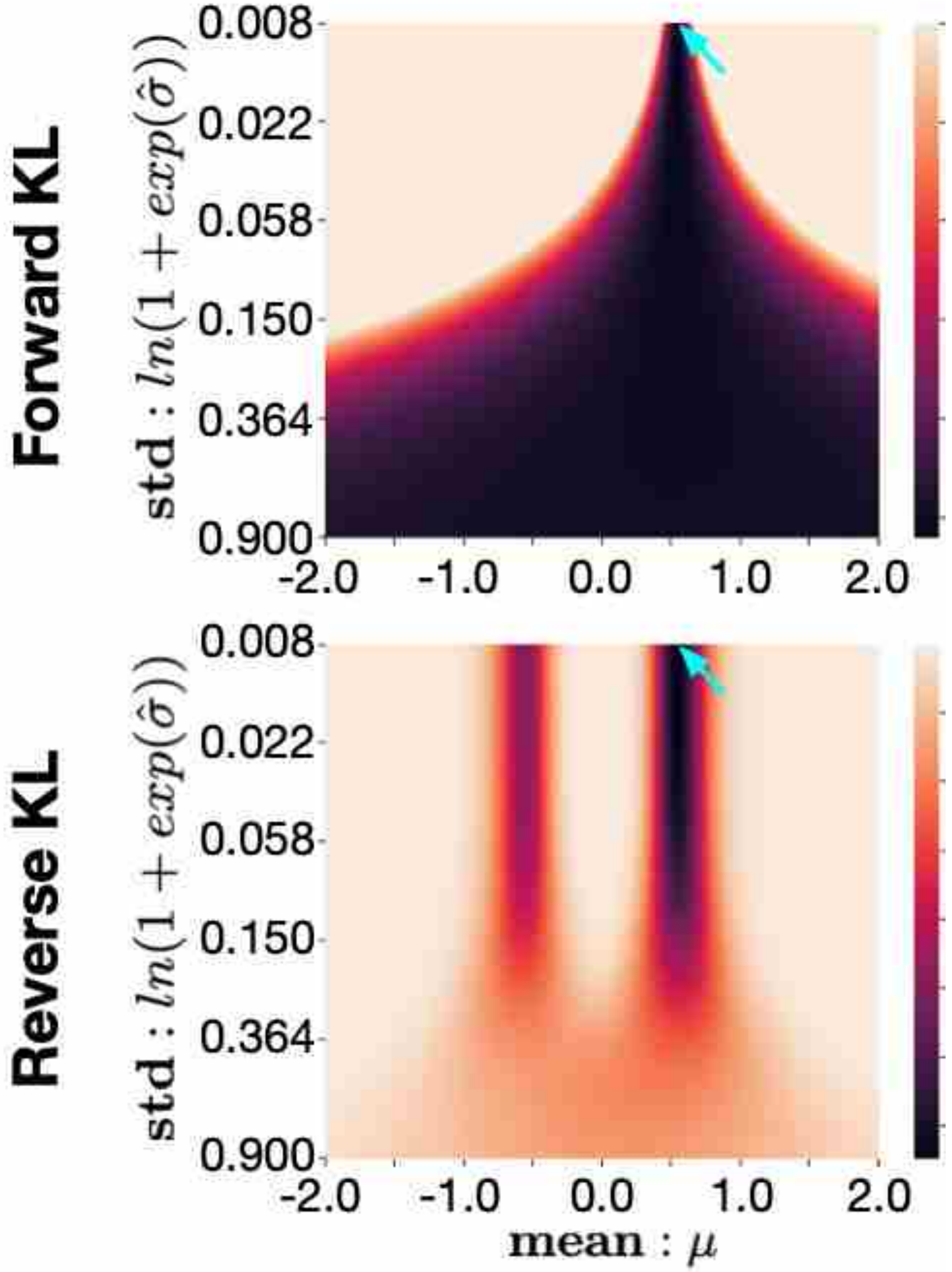
Forward vs Reverse KL and convexity

- **Forward KL**: $\text{KL}(\pi_{\text{ent}}(\cdot \mid s) \,||\, \pi_\theta(\cdot \mid s))$
  - convex for Boltzmann policies
- **Reverse KL**: $\text{KL}(\pi_\theta(\cdot \mid s) \,||\, \pi_{\text{ent}}(\cdot \mid s))$
  - non-convex even for nice distributions

* See our journal paper: “Greedification Operators for Policy Optimization: Investigating Forward and Reverse KL Divergences”, Chan et al., JMLR, 2022

Forward vs Reverse KL and convexity

- **Forward KL**: KL$(\pi_{\text{ent}}(\cdot \mid s) \,||\, \pi_\theta(\cdot \mid s))$
  - convex for Boltzmann policies
- **Reverse KL**: KL$(\pi_\theta(\cdot \mid s) \,||\, \pi_{\text{ent}}(\cdot \mid s))$
  - non-convex even for nice distributions



* See our journal paper: "Greedification Operators for Policy Optimization: Investigating Forward and Reverse KL Divergences", Chan et al., JMLR, 2022

Motivates reconsidering local updates that can get stuck

And exploring alternatives

Next

- Explain our GreedyAC algorithm, inspired by this motivation

Next

- Explain our GreedyAC algorithm, inspired by this motivation
- Work lead by PhD student Samuel Neumann

* See our paper: “Greedy Actor-Critic: A New Conditional Cross-Entropy Method for Policy Improvement”, Neumann et al., ICLR, 2023

A Brief Interlude about CEM

Goal: find $\arg\max_{\theta} f(\theta)$

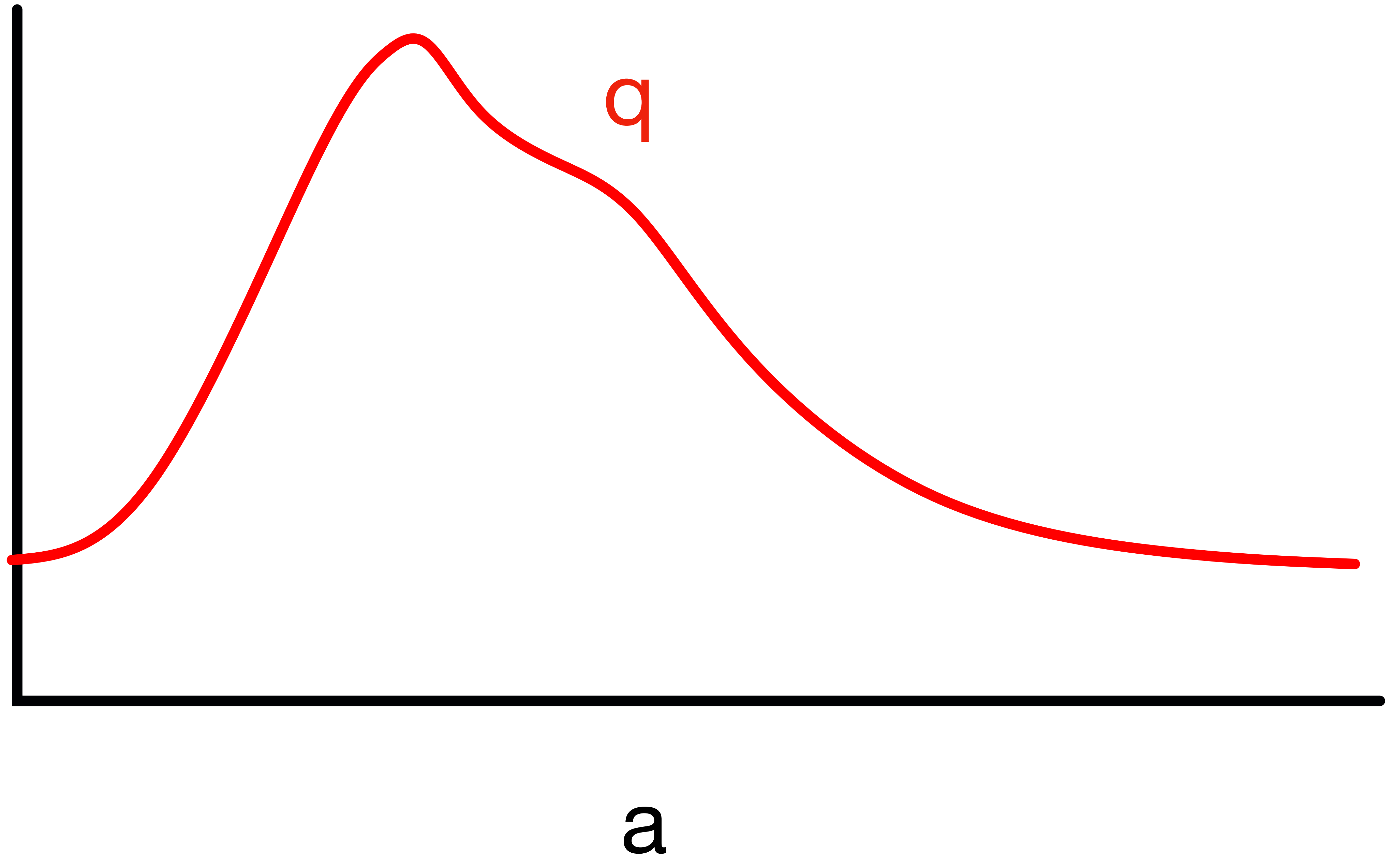
A Brief Interlude about CEM

Goal: find $\arg\max_a q(a)$

A Brief Interlude about CEM

Goal:

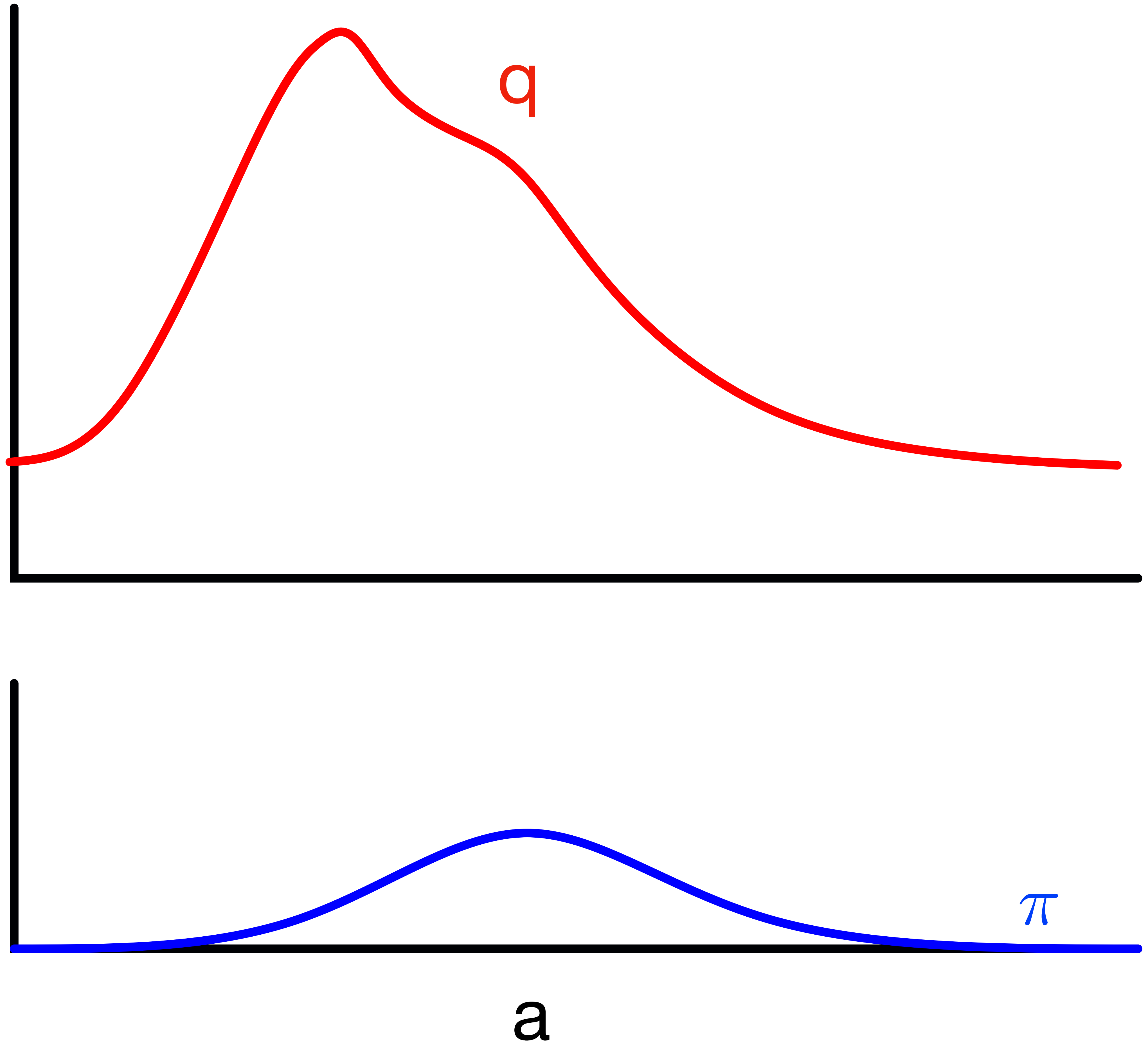
find $\arg\max_{a} q(a)$



A Brief Interlude about CEM

Goal:
find $\arg\max_a q(a)$

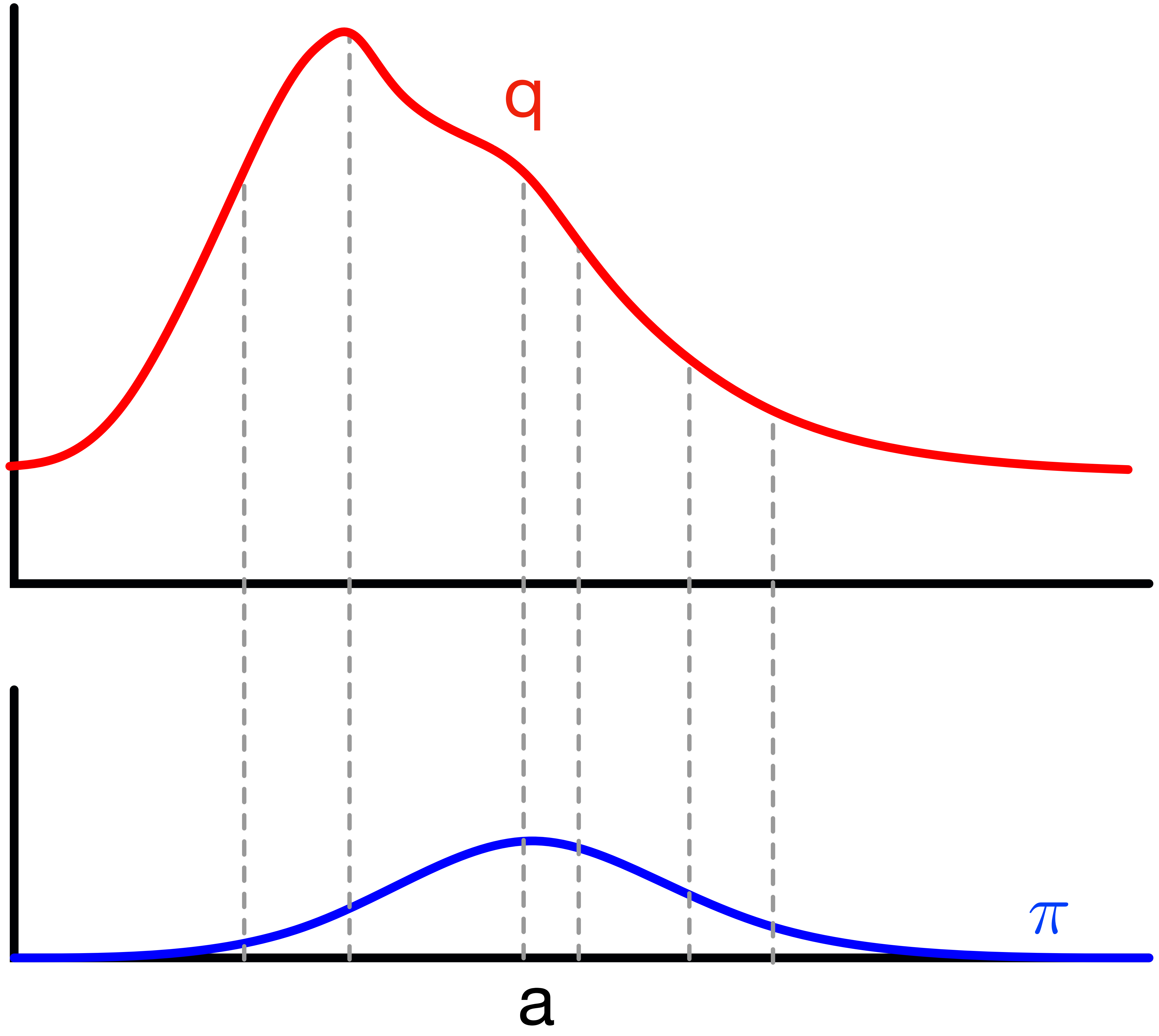
Introduce distribution π that concentrates on maximal a



CEM in Action

Goal:
find $\arg\max_a q(a)$

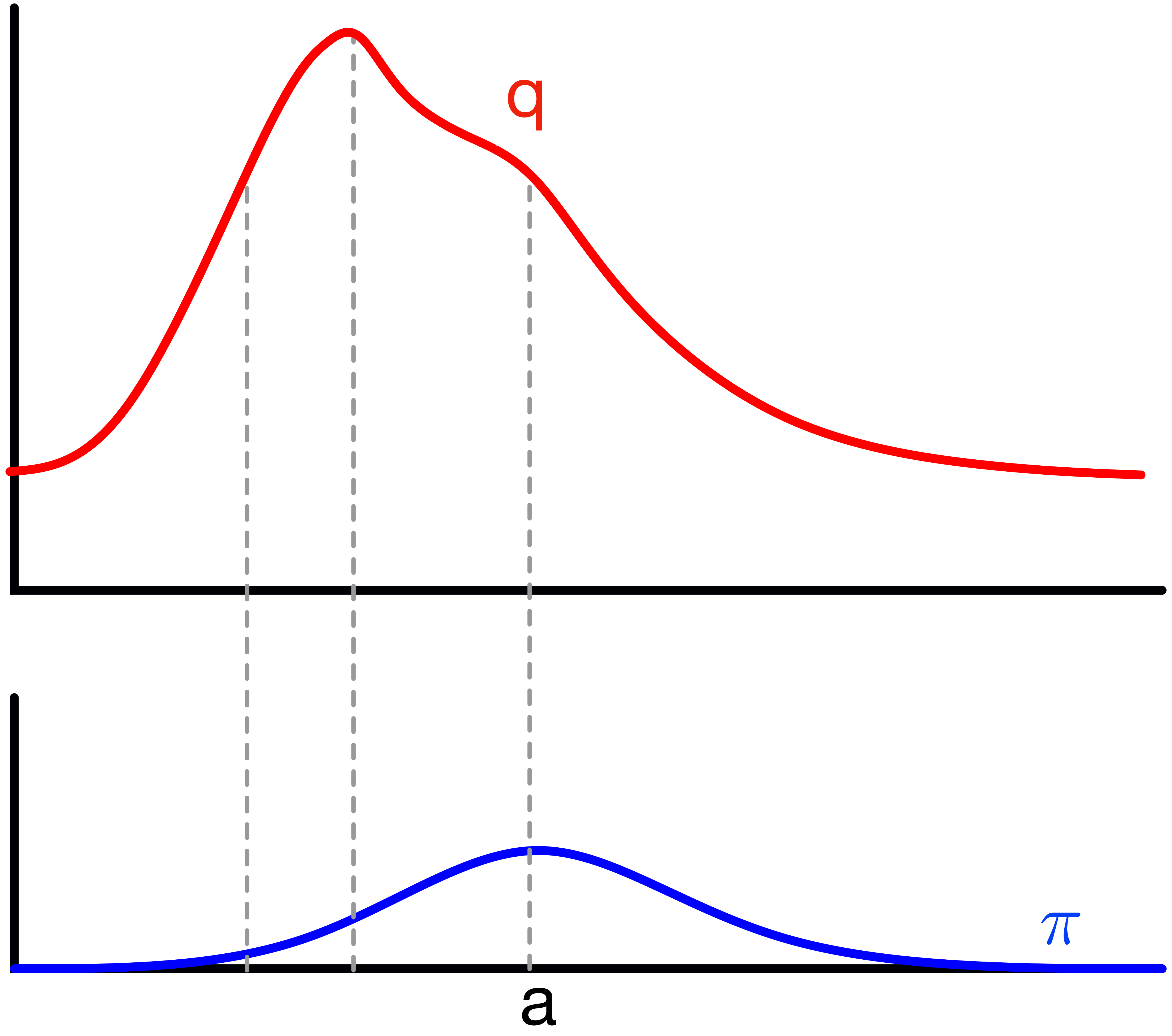
Sample a from π



CEM in Action

Goal:
find $\arg\max_a q(a)$

Take top percentile
according to $q(a)$

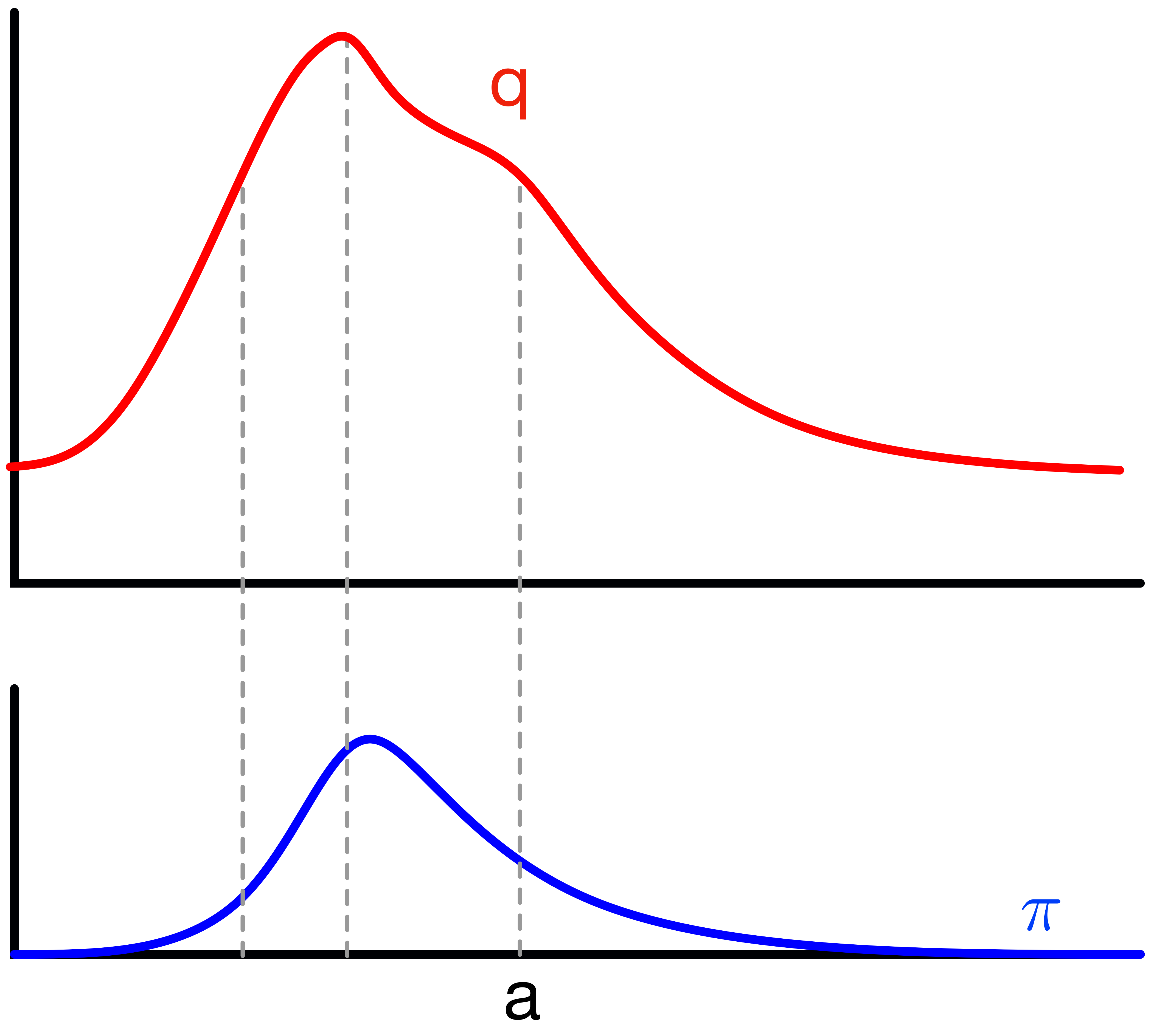


CEM in Action

Goal:

find $\arg\max_{a} q(a)$

Increase likelihood of a in top percentile



We want $\pi(a \mid s)$ to concentrate on top actions of $q(s, a)$
Like CEM, but now conditioned on states

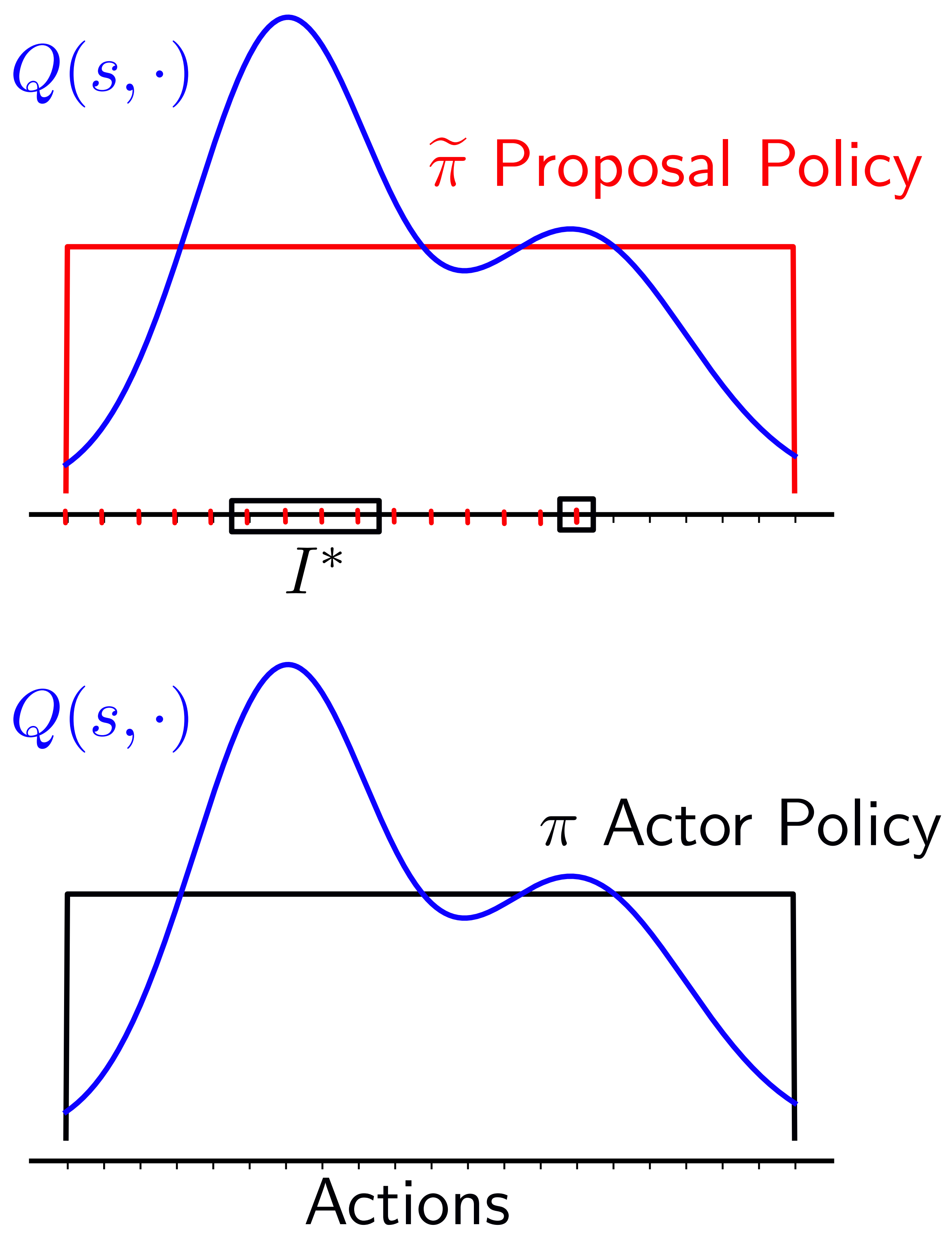
Conditional CEM Algorithm

- Assume **action-values** q are fixed and given, for now
- Learn **actor policy** $\pi(a \mid s)$ that gradually increase likelihood of top actions, across states

Conditional CEM Algorithm

- Assume **action-values** q are fixed and given, for now
- Learn **actor policy** $\pi(a \mid s)$ that gradually increase likelihood of top actions, across states
- **Issue**: $\pi(a \mid s)$ will likely concentrate too quickly, before seeing all states
  - i.e., we can't just apply the exact same idea as CEM naively
- **Fix**: introduce a more slowly changing **proposal policy** $\tilde{\pi}(a \mid s)$

Conditional CEM in Action



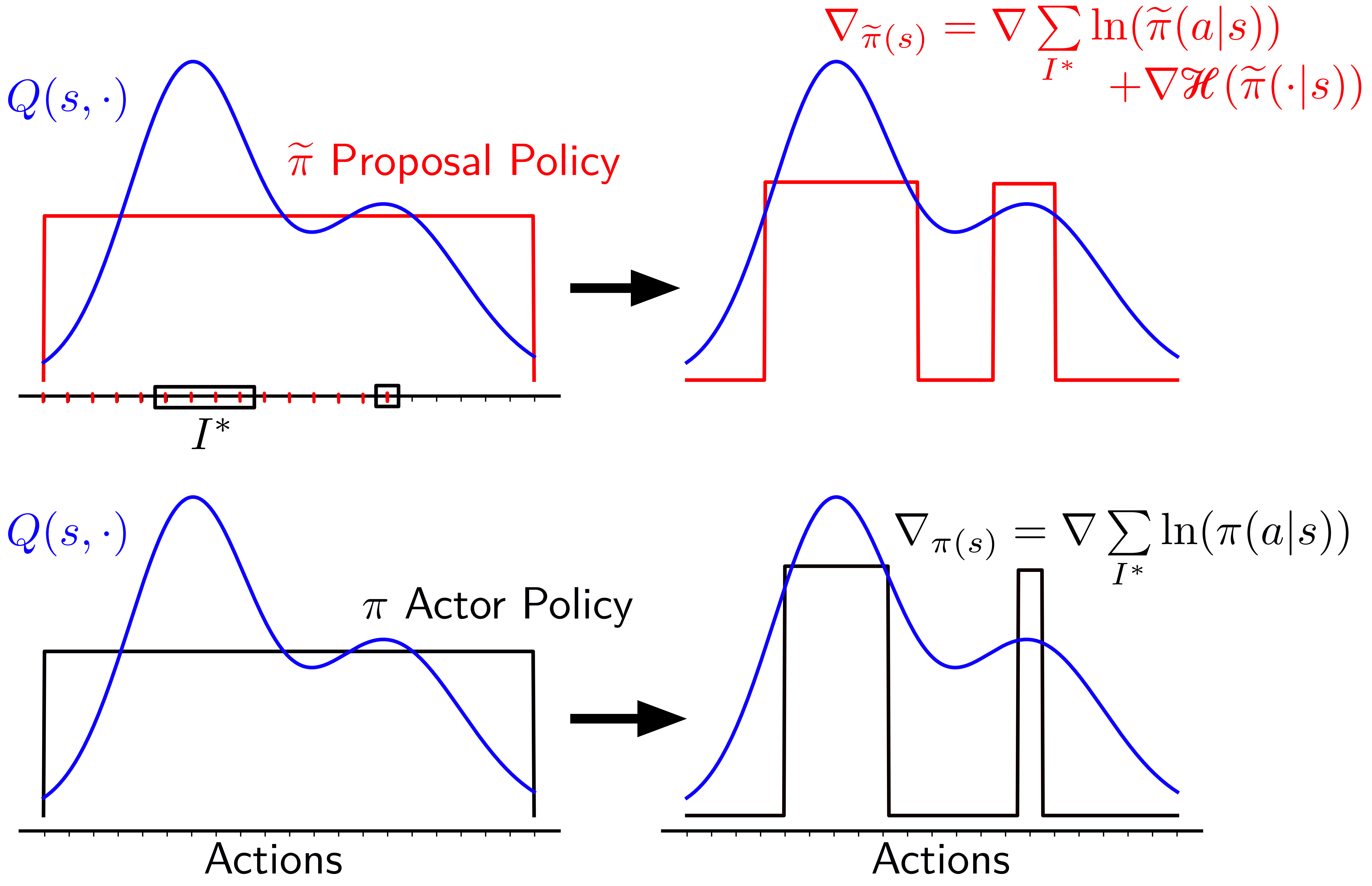
Sample a state s (or mini-batch of states)

Sample 15 actions $a_1, a_2, \ldots, a_{15} \sim \tilde{\pi}(\cdot \mid s)$

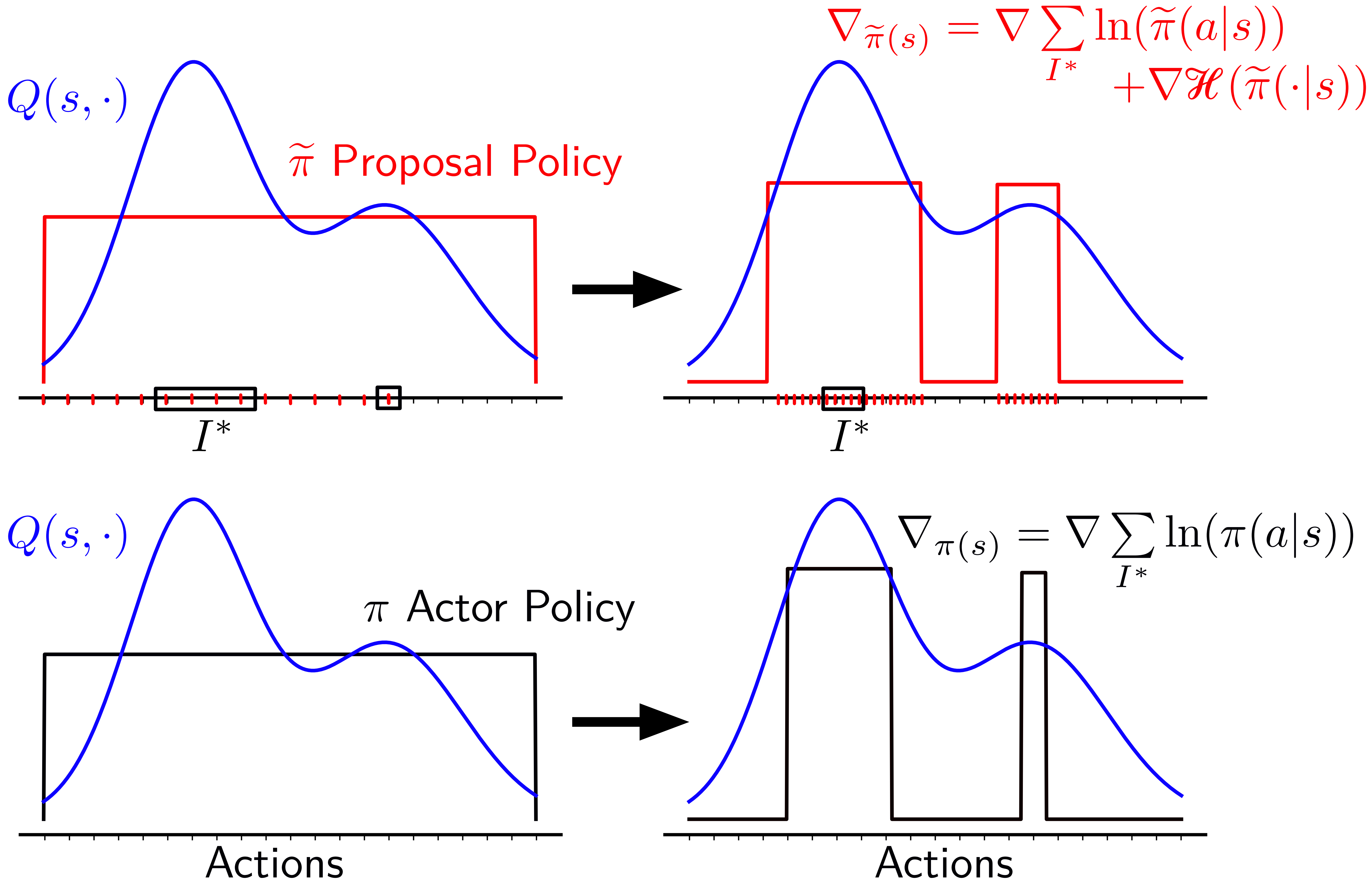
* Note: we do not actually use a uniform distribution for the policies, it is just easier to visualize here in this example

Identify $I^* =$ top 5 (namely the 0.33 percentile)

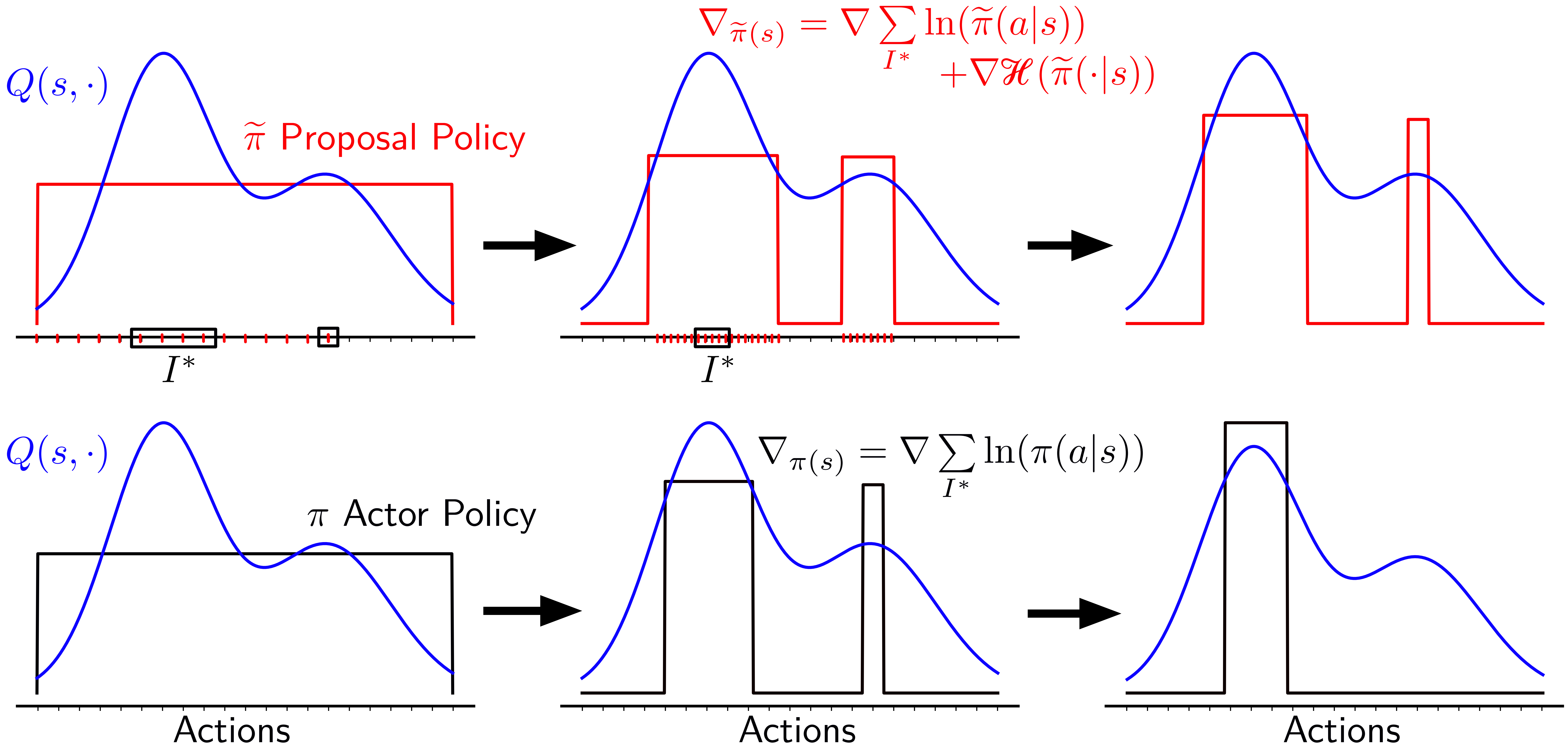
Conditional CEM in Action



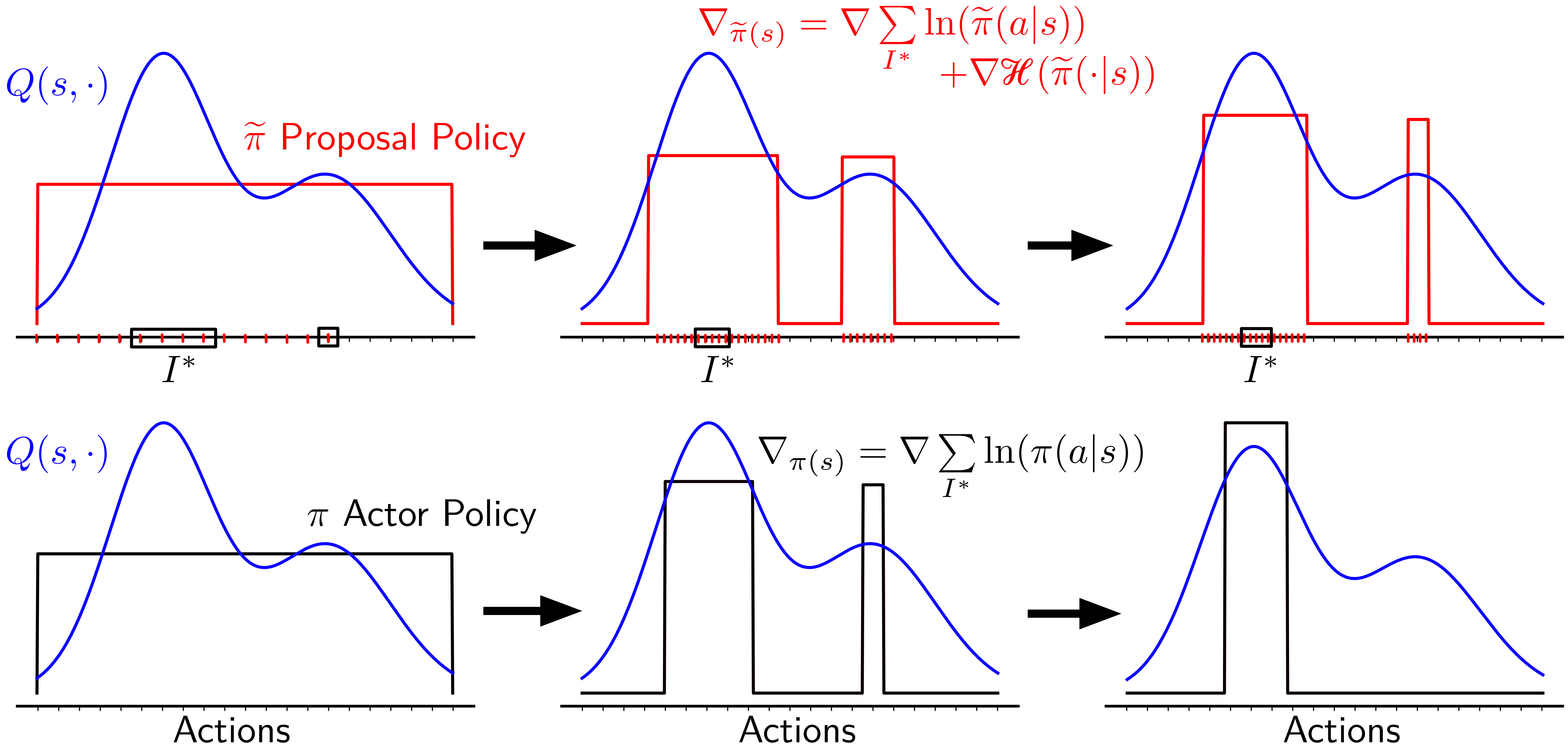
Conditional CEM in Action



Conditional CEM in Action



Conditional CEM in Action



Theory for why we have two policies

- Two timescale analysis:
  - q and $\tilde{\pi}$ changing at a slower timescale, so we can consider them fixed when analyzing the update for the actor π
- Result says updates behaves like CEM, in expectation across states
- Tracks the CEM update, as q (slowly) changes

Policy Improvement Guarantees

- Log-likelihood update to π corresponds to minimizing a forward KL to a percentile policy
  - KL $\left(\pi_{\text{percentile}}(\cdot \mid s) \,||\, \pi(\cdot \mid s)\right)$
- **Percentile policy on q_π guaranteed to be a better policy**
  - namely $\mathbb{E}_{a\sim\pi'}[q_{\pi'}(s,a)] \geq \mathbb{E}_{a\sim\pi}[q_\pi(s,a)]$ for $\pi' = \pi_{\text{percentile}}$

Policy Improvement Guarantees

- Log-likelihood update to π corresponds to minimizing a forward KL to a percentile policy
  - KL $\left(\pi_{\text{percentile}}(\cdot \mid s) \mid\mid \pi(\cdot \mid s)\right)$
- Percentile policy on q_π guaranteed to be a better policy
  - namely $\mathbb{E}_{a \sim \pi'}[q_{\pi'}(s, a)] \geq \mathbb{E}_{a \sim \pi}[q_\pi(s, a)] \quad$ for $\pi' = \pi_{\text{percentile}}$
- We named the algorithm **GreedyAC** because it eventually concentrates on the greedy actions (unregularized), unlike Soft Actor-Critic

Contrasting to SAC and other AC methods

- GreedyAC uses: KL $\left(\pi_{\text{percentile}}(\cdot \mid s) \,||\, \pi(\cdot \mid s)\right)$
- Most AC methods minimize a reverse KL to π_{ent} or π_{kl}
  - KL $\left(\pi(\cdot \mid s) \,||\, \pi_{\text{ent}}(\cdot \mid s)\right)$ or KL $\left(\pi(\cdot \mid s) \,||\, \pi_{\text{kl}}(\cdot \mid s)\right)$

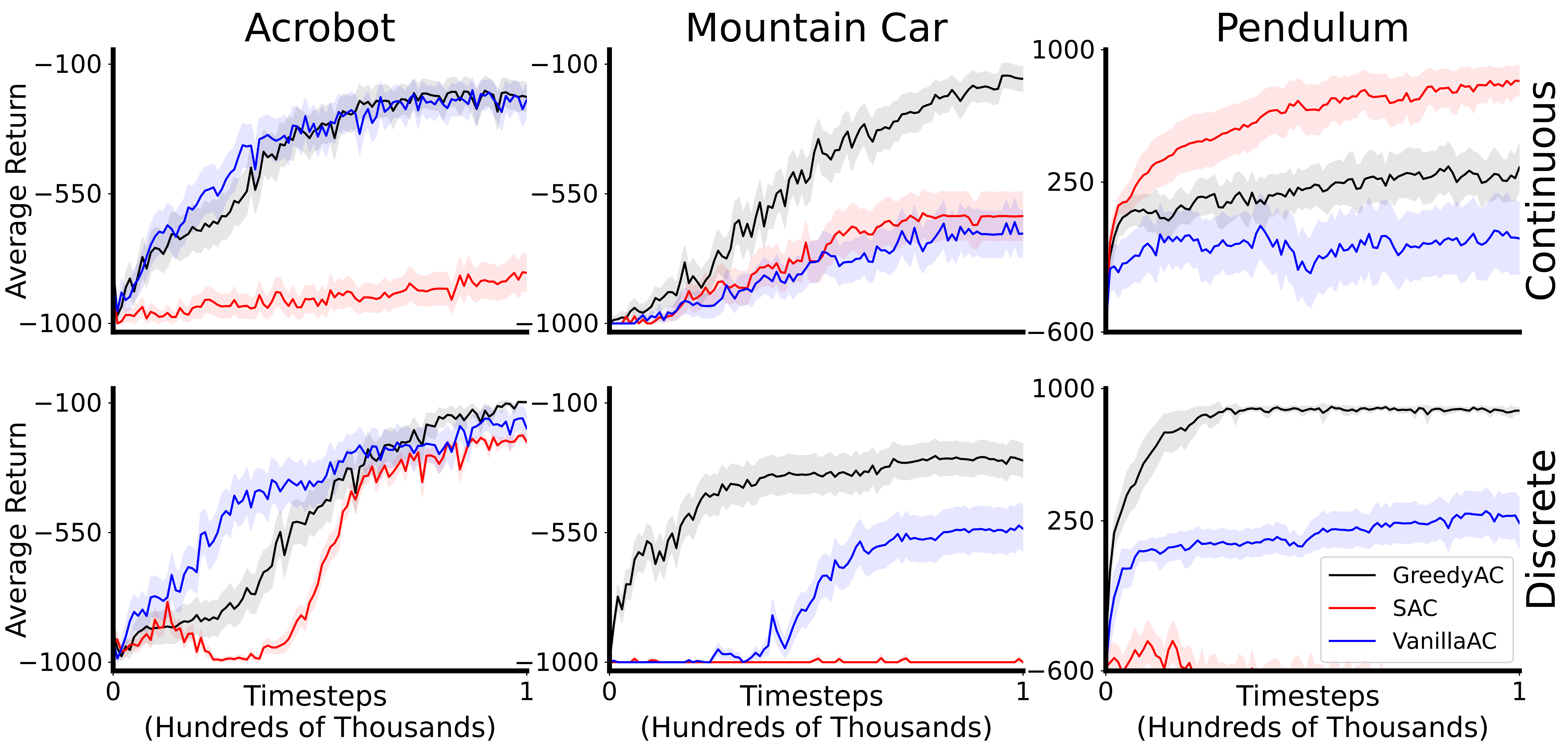
Similarity to MPO

- GreedyAC uses: KL $\left(\pi_{\text{percentile}}(\cdot \mid s) \mid\mid \pi(\cdot \mid s)\right)$
- MPO minimizes a forward KL to π_{kl}, by increasing likelihood of actions sampled from π_{kl}
    - KL $\left(\pi_{\text{kl}}(\cdot \mid s) \mid\mid \pi(\cdot \mid s)\right)$

Back to our simple classic control environments

- All agents use neural networks, the Adam optimizer, and replay

*entropy, critic & actor stepsize tuned **across** environments



*more results in the paper, on MinAtari and Swimmer from Mujoco

Why might GreedyAC be better than SAC?

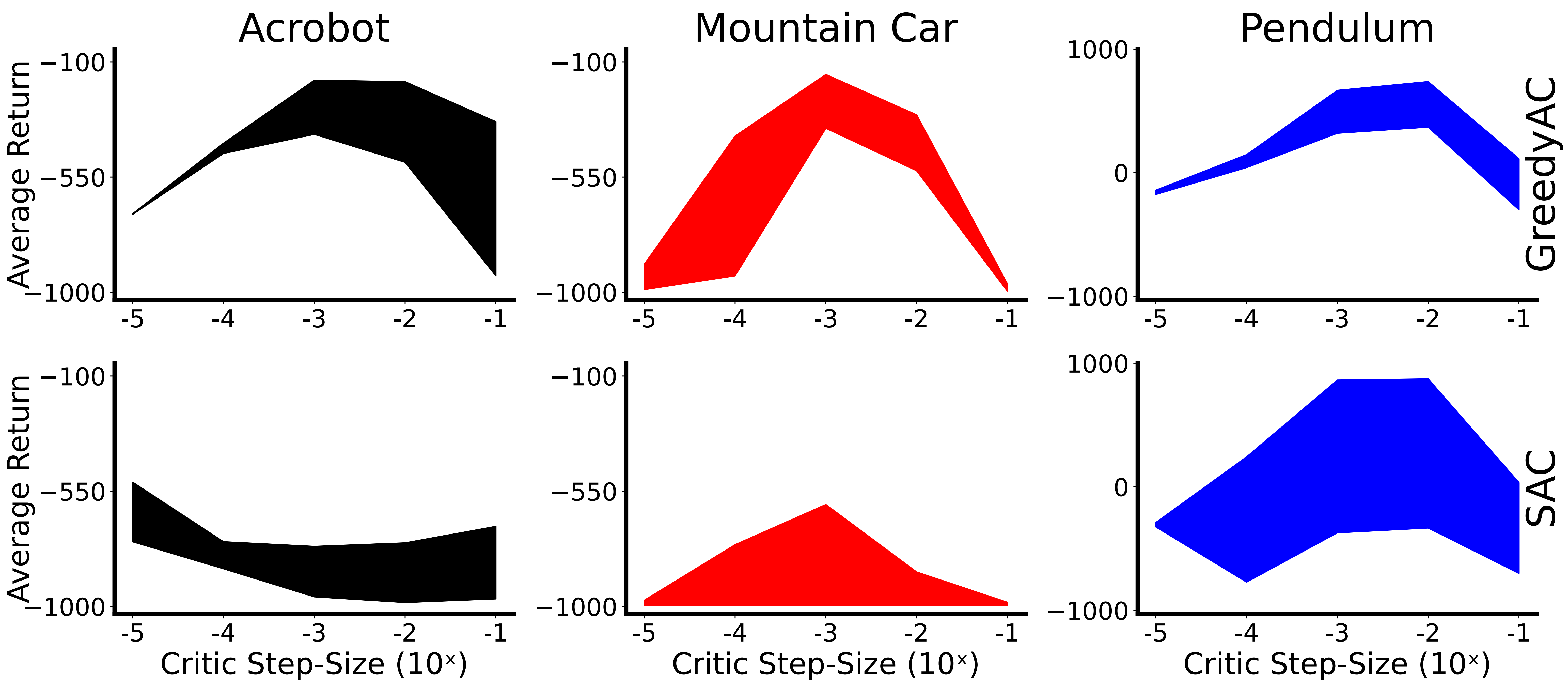
- SAC is sensitive to its entropy parameter
- Entropy potentially plays many roles in SAC
  - prevents policy collapse, promotes exploration, smoothing the objective

Why might GreedyAC be better than SAC?

- SAC is sensitive to its entropy parameter
- Entropy potentially plays many roles in SAC
    - prevents policy collapse, promotes exploration, smoothing the objective
- GreedyAC only uses the entropy to slow the concentration of the proposal policy (one role)

Understanding sensitivity to entropy

- Solid area is range of performance across different entropy values
- Wider is bad
- Lower is bad



Conclusions

- Finicky behavior of actor-critic methods might be due to interacting choices
  - did not reweight states, did not get critic error low enough, did not do enough greedification, or did not avoid changing the policy too much…

Conclusions

- Finicky behavior of actor-critic methods might be due to interacting choices
  - did not reweight states, did not get critic error low enough, did not do enough greedification, or did not avoid changing the policy too much…
- Initial results for GreedyAC look promising as a simpler actor update
  - intuitive percentile parameter, does not rely on entropy
  - one part of this puzzle, we are continuing to work on interacting choices

Conclusions

- Finicky behavior of actor-critic methods might be due to interacting choices
  - did not reweight states, did not get critic error low enough, did not do enough greedification, or did not avoid changing the policy too much*…
- Initial results for GreedyAC look promising as a simpler actor update
  - intuitive percentile parameter, does not rely on entropy
  - one part of this puzzle, we are continuing to work on interacting choices*

* RLC 2025 paper “**Investigating the Utility of Mirror Descent in Off-policy Actor-Critic**”

Conclusions

- Finicky behavior of actor-critic methods might be due to interacting choices
    - did not reweight states, did not get critic error low enough, did not do enough greedification, or did not avoid changing the policy too much…
- Initial results for GreedyAC look promising as a simpler actor update
    - intuitive percentile parameter, does not rely on entropy
    - one minor part of this puzzle, we are working on interacting choices
- This is an exciting time to be making better actor-critics
    - lots of theoretical insights, more can make its way into practice
    - lots to understand empirically about the sea of algorithms

Questions?