10-403: Maximum Entropy RL
Formulation & Algorithms

Instructors: Aviral Kumar & Katerina Fragkiadaki

Carnegie Mellon University
School of Computer Science

Lecture Outline

- **Maximum Entropy RL formulation**
- **Maximum Entropy RL Algorithms:**
  - Maximum entropy policy gradient
  - Soft Actor-Critic
    - Related algorithms (DDPG, TD3)
- **Where Does Maximum Entropy RL Help?**
  - Robust Solutions in Precision-Heavy Problems
  - Smoothens Out the Optimization Dynamics
  - Exploration in a Structured Way

Formulation: Maximum Entropy RL

$$\pi^* := \arg\max_{\pi} \quad \mathbb{E}_{\tau \sim \pi} \left[\sum_t \gamma^t r(s_t, a_t) \right]$$

$$\pi^*_{\text{maxent}} := \arg\max_{\pi} \quad \mathbb{E}_{\tau \sim \pi} \left[\sum_t \gamma^t r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t) \right]$$

Properties of Maximum Entropy Policies

$$\pi^*_{\text{maxent}} := \arg\max_{\pi} \quad \mathbb{E}_{\tau\sim\pi}\left[\sum_t \gamma^t r(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t)\right]$$

- The maximum entropy policy is stochastic. Whereas there is always a deterministic optimal policy in the MDP.
- Stochasticity in a policy → noise during exploration when learning
- Want to put equal probability mass on different ways of solving the task *(principle of maximum entropy)*

Optimizing the Maximum Entropy RL Objective

$$\pi^*_{\text{maxent}} := \arg\max_{\pi} \quad \mathbb{E}_{\tau\sim\pi}\left[\sum_t \gamma^t r(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t)\right]$$

$$\pi^*_{\text{maxent}} := \arg\max_{\pi} \quad \mathbb{E}_{\tau\sim\pi}\left[\sum_t \gamma^t r(s_t, a_t) + \alpha\mathbb{E}_{a\sim\pi(\cdot|s_t)}\left[-\log\pi(a|s_t)\right]\right]$$

$$\pi^*_{\text{maxent}} := \arg\max_{\pi} \quad \mathbb{E}_{\tau\sim\pi}\left[\sum_t \gamma^t r(s_t, a_t) - \alpha\log\pi(a_t|s_t)\right]$$

Optimizing the Maximum Entropy RL Objective

Why is this not entropy-regularized PG?

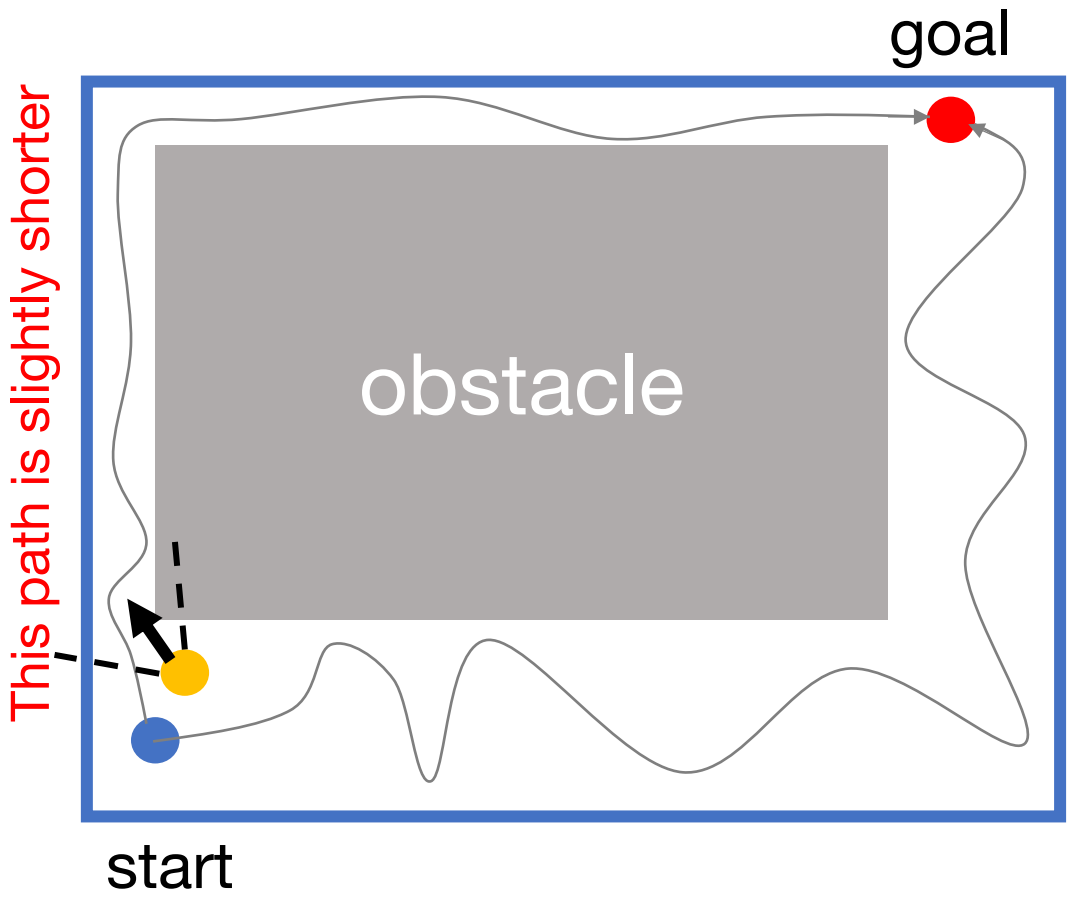
Optimizing the Maximum Entropy RL Objective

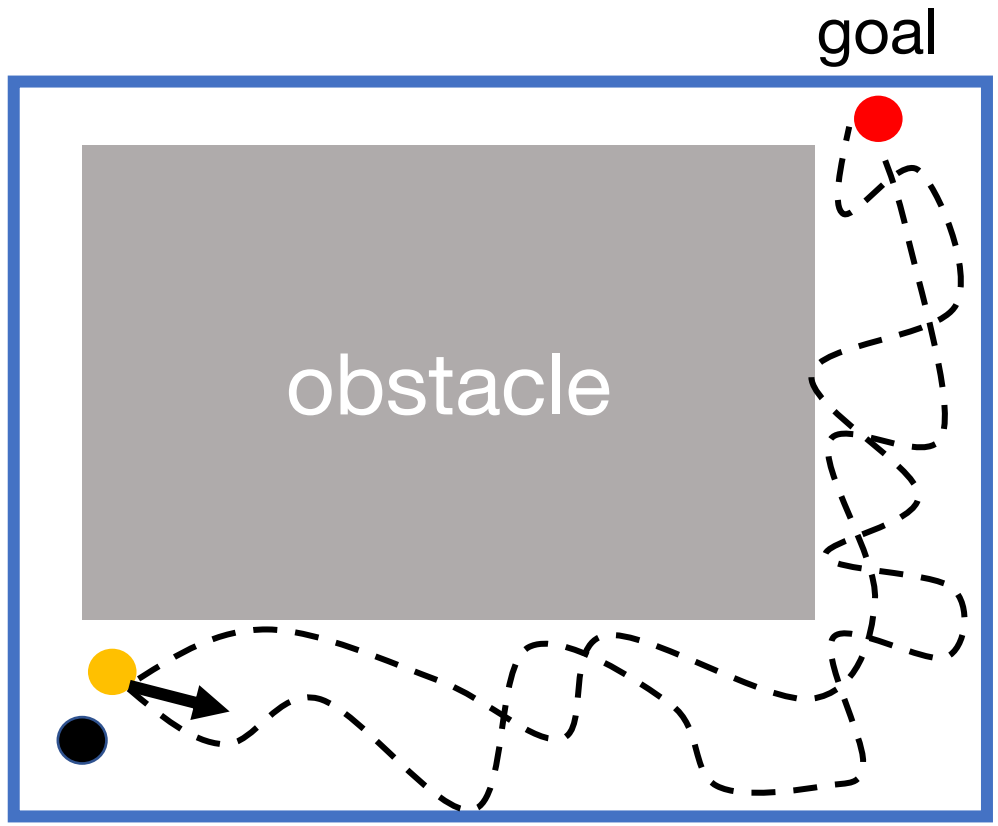
Why is this not entropy-regularized PG?

Answer: *Backing up the entropy*

Backing up vs Not Backing up Entropy

- The difference emerges from whether you want the policy to take you to states from which you can succeed even if you are noisy or not





More ways to succeed when entropy is backed up!

Optimizing via Soft Bellman Equations

$$Q^{\pi}(s,a) := r(s,a) + \gamma \mathbb{E}_{s' \sim p(\cdot|s,a), a' \sim \pi(\cdot|s')}[Q^{\pi}(s',a')]$$

Definition: The total reward attained by following the action a at state s, followed by following \pi thereafter

$$Q^{\pi}_{\text{soft}}(s,a) := r(s,a) + \gamma \mathbb{E}_{s',a' \sim \pi(\cdot|s')}\left[Q^{\pi}_{\text{soft}}(s',a') - \alpha \log \pi(a'|s')\right]$$

Definition: The total reward attained by following the action a at state s, followed by backing Q_{soft} and entropy thereafter

Haarnoja et al. **Reinforcement Learning with Deep Energy Based Policies**. ICML 2017.

Optimizing via Soft Bellman Equations

Are there other ways of writing the backup?

***Answer*:** *Entropy can go to s or s'?*

Contraction of the Soft Bellman Equations

Definition: The total reward attained by following the action a at state s, followed by backing Q_{soft} and entropy thereafter

$$Q^{\pi}_{\text{soft}}(s,a) := r(s,a) + \gamma \mathbb{E}_{s',a' \sim \pi(\cdot|s')} \left[Q^{\pi}_{\text{soft}}(s',a') - \alpha \log \pi(a'|s') \right]$$

$$Q^{\pi}_{\text{soft}}(s,a) := \left(r(s,a) - \alpha \mathbb{E}_{s',a' \sim \pi(\cdot|s')} [\log \pi(a'|s')] \right) + \gamma \mathbb{E}_{s',a' \sim \pi(\cdot|s')} \left[Q^{\pi}_{\text{soft}}(s',a') \right]$$

New ”effective” reward; proof of convergence is same as standard Bellman

Haarnoja et al. **Reinforcement Learning with Deep Energy Based Policies**. ICML 2017.

Algorithm: Soft Policy Iteration

Algorithm: ~~Soft~~ Policy Iteration

Policy evaluation: *Evaluate the Q-function of the policy by running Bellman backups multiple times*

$$Q(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')} \left[Q_{\text{target}}(s', a') \right]$$

Policy improvement: *Update the policy towards argmax Q*

$$\forall \ s, \quad \pi'(\cdot|s) := \arg\max_{a} Q(s,a)$$

Algorithm: Soft Policy Iteration

Soft policy evaluation: *Evaluate the soft Q-function of the policy by running Bellman backups multiple times*

$$Q(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')} \left[Q_{\text{target}}(s', a') - \alpha \log \pi(a'|s') \right]$$

Soft policy improvement: *Update the policy towards maximizing the Q-value, regularized with entropy*

$$\forall\ s, \quad \pi'(\cdot|s) := \arg\max_{\pi} \mathbb{E}_{a \sim \pi(\cdot|s)} \left[Q(s,a) - \alpha \log \pi(a|s) \right]$$

Algorithm: Soft Policy Iteration

Soft policy evaluation: *Evaluate the soft Q-function of the polic… times*

$Q(s,a) \leftarrow r \ldots \log \pi(a'|s')]$

Soft policy … *…ards softmax*

$\forall\ s,\quad \pi'(\cdot|s) := \arg\max_{\pi} \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q(s,a) - \alpha \log \pi(a|s)\right]$

What would the algorithm look like if the entropy were at s?

Implication: Matching the SoftMax Optimal Policy

$$\forall\ s,\quad \pi'(\cdot|s) := \arg\max_{\pi} \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q(s,a) - \alpha \log \pi(a|s)\right]$$

Does this expression look familiar? Can we write this term as a KL divergence?

$$\forall\ s,\quad \pi'(\cdot|s) := \arg\min_{\pi}\ \mathbb{E}\left[\log \pi(a|s) - \log \frac{\exp(Q(s,a)}{Z(s)} + \log \cancel{Z(s)}\right]$$

"Softmax" optimal policy

$$\forall\ s,\quad \pi'(\cdot|s) := \arg\min_{\pi}\ \mathbb{E}_s\left[\mathrm{D_{KL}}\left(\pi(\cdot|s)||\frac{\exp(Q(s,a))}{Z(s)}\right)\right]$$

Implication: Matching the SoftMax Optimal Policy

Can prove that soft Q-values increase monotonically!

Practical Algorithms: DDPG vs Soft Actor-Critic

DDPG

1. Policy evaluation:

$$y = r(s,a) + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q_{\bar{\theta}}(s', a')]$$

$$\min_{\theta} \mathbb{E}_{s,a,s'} \left[(Q_\theta(s,a) - y)^2 \right]$$

2. Policy improvement:

$$\max_{\phi} \mathbb{E}_s \left[Q(s, a = \pi_\phi(s)) \right]$$

SAC

1. Policy evaluation:

$$y = r(s,a) + \gamma E_{\pi(\cdot|s')}[Q(s', a') - \alpha \log \pi(a'|s')]$$

$$\min_{\theta} \mathbb{E}_{s,a,s'} \left[(Q_\theta(s,a) - y)^2 \right]$$

2. Policy improvement:

$$\max_{\phi} \mathbb{E}_{s,a \sim \pi_\phi(\cdot|s)}[Q(s,a) - \alpha \log \pi_\phi(a|s)]$$

Lillicrap et al. **Continuous control with deep reinforcement learning.** ICLR 2015.
Haarnoja et al. Soft Actor-Critic: Algorithms and Applications. arXiv 2018.

Practical Algorithms: DDPG vs Soft Actor-Critic

DDPG

3. Exploration

$$\epsilon_0 \sim \mathcal{N}(0, 1)$$

$$\epsilon_{t+1} = (1 - \theta)\epsilon_t + \sigma\mathcal{N}(0, 1)$$

Ornstein-Uhlenbeck (OU) process

$$a_{t+1} = \pi(s_{t+1}) + \epsilon_{t+1}$$

SAC

3. Exploration

Simply sample from the stochastic policy!

$$a_{t+1} \sim \pi_\phi(\cdot|s_{t+1})$$

But we run deterministic policy at evaluation time.

Lillicrap et al. **Continuous control with deep reinforcement learning.** ICLR 2015.
Haarnoja et al. **Soft Actor-Critic: Algorithms and Applications.** arXiv 2018.

Practical Algorithms: DDPG

Algorithm 1 Deep Deterministic Policy Gradient

Input: initial policy parameters θ, Q-function parameters ϕ, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\theta_{\text{targ}} \leftarrow \theta$, $\phi_{\text{targ}} \leftarrow \phi$
repeat
  Observe state s and select action $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{Low}, a_{High})$, where $\epsilon \sim \mathcal{N}$
  Execute a in the environment
  Observe next state s', reward r, and done signal d to indicate whether s' is terminal
  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
  If s' is terminal, reset environment state.
  if it's time to update **then**
    for however many updates **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute targets

$$y(r, s', d) = r + \gamma(1 - d)Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))$$

      Update Q-function by one step of gradient descent using

$$\nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_\phi(s, a) - y(r, s', d)\right)^2$$

      Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_\phi(s, \mu_\theta(s))$$

      Update target networks with

$$\phi_{\text{targ}} \leftarrow \rho\phi_{\text{targ}} + (1 - \rho)\phi$$
$$\theta_{\text{targ}} \leftarrow \rho\theta_{\text{targ}} + (1 - \rho)\theta$$

See "clipping" actions to be in bounds

Replay buffer like DQN

Compute Q-targets by querying target network

Run gradient step on TD-loss

Take the gradient of the policy w.r.t. the Q-function

SAC: Practical Implementation Details

- **Clipped Double Q-Learning** (aka TD3 trick, Fujimoto et al. 2018)

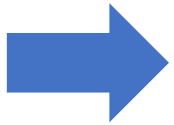
$$y = r(s,a) + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')} \left[\min_{i=1,2} Q_{\bar{\theta}}(s', a') - \alpha \log \pi(a'|s') \right]$$

Minimum of two Q-functions; this helps in reducing overestimation.
Closely related to double Q-learning, but this seems to work better.

- **Automatic entropy tuning**

$$\mathbb{E}\left[\sum_t \gamma^t r(s_t, a_t) + \alpha \mathcal{H}[\pi(\cdot|s_t)] \right]$$

Now this varies through training!



$$\max \quad \mathbb{E}\left[\sum_t \gamma^t r(s_t, a_t) \right]$$

$$\text{s.t.} \quad \mathbb{E}\left[\mathcal{H}[\pi(\cdot|s_t)] \right] \geq \varepsilon$$

SAC: Practical Implementation Details

- **Reparameterization policy gradient**

$$\nabla_\phi \text{loss}(\phi) = \nabla_\phi \mathbb{E}_{s_t} \mathbb{E}_{a_t \sim \pi_\phi(a|s_t)} \log \frac{\pi_\phi(a_t|s_t)}{\exp(Q_\theta(s_t, a_t))}$$

Typically need to use policy gradient here, but say my policy is….

$$a_t = f_\phi(s_t, \epsilon) = \mu_\phi(s_t) + \epsilon \Sigma_\phi(s_t), \quad \epsilon \sim \mathcal{N}(0, I)$$

Free of policy parameters!

$$\nabla_\phi \text{loss}(\phi) = \nabla_\phi \mathbb{E}_{s_t, \epsilon \sim \mathcal{N}(0,I)} \log \frac{\pi_\phi(a_t|s_t)}{\exp(Q_\theta(s_t, a_t))}$$

Algorithm 1 Soft Actor-Critic

Input: initial policy parameters θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$
repeat
  Observe state s and select action $a \sim \pi_\theta(\cdot|s)$
  Execute a in the environment
  Observe next state s', reward r, and done signal d to indicate whether s' is terminal
  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
  If s' is terminal, reset environment state.
  if it's time to update **then**
    for j in range(however many updates) **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute targets for the Q functions:

$$y(r, s', d) = r + \gamma(1 - d)\left(\min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')\right), \quad \tilde{a}' \sim \pi_\theta(\cdot|s')$$

      Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_{\phi_i}(s, a) - y(r, s', d)\right)^2 \qquad \text{for } i = 1, 2$$

      Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta\left(\tilde{a}_\theta(s)|\, s\right)\right),$$

      where $\tilde{a}_\theta(s)$ is a sample from $\pi_\theta(\cdot|s)$ which is differentiable wrt θ via the reparametrization trick.
      Update target networks with

$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho)\phi_i \qquad \text{for } i = 1, 2$$

Sampling actions from the policy

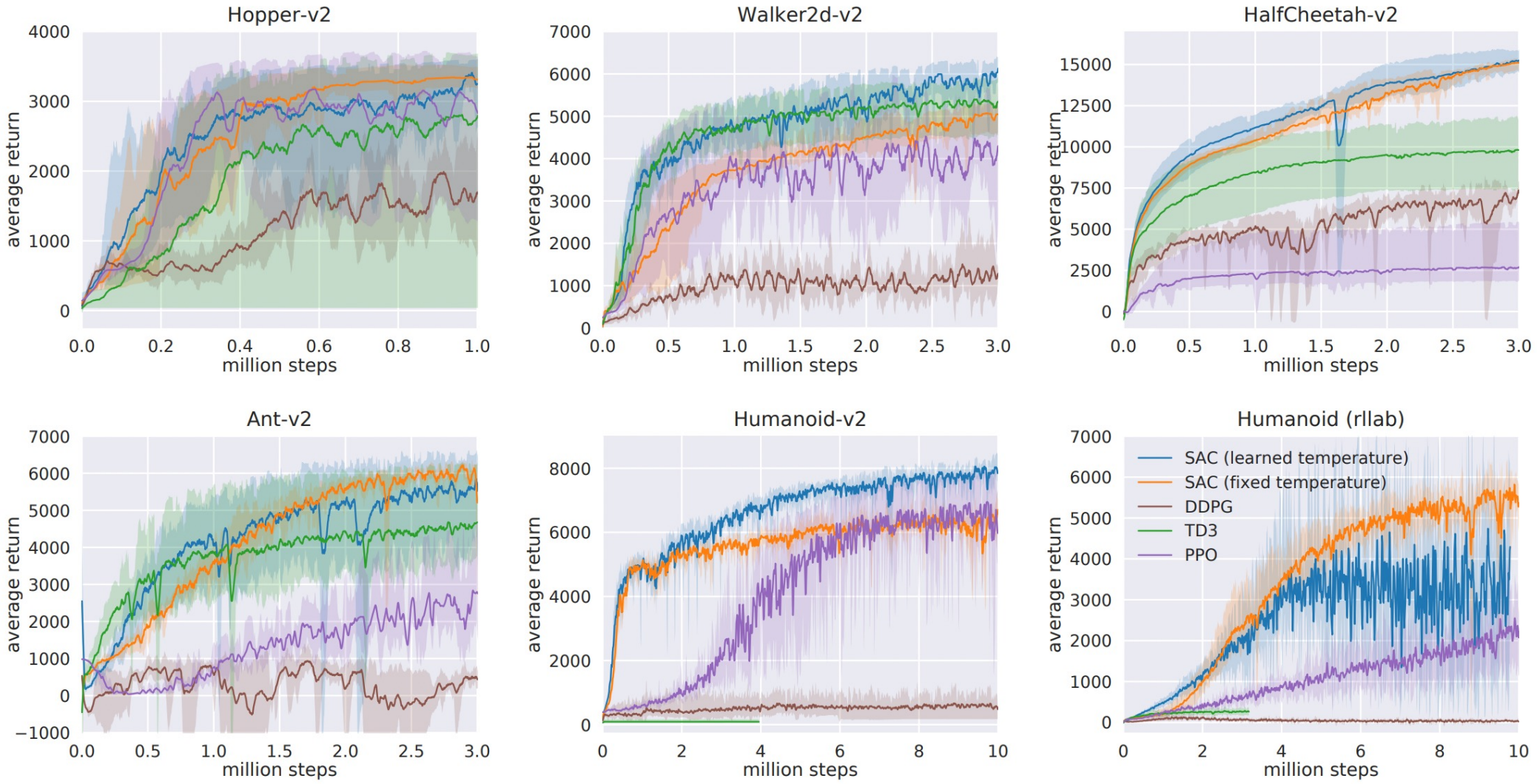
Store data in the replay buffer

Compute target values with the "min-Q" trick

Train Q-values via TD-learning

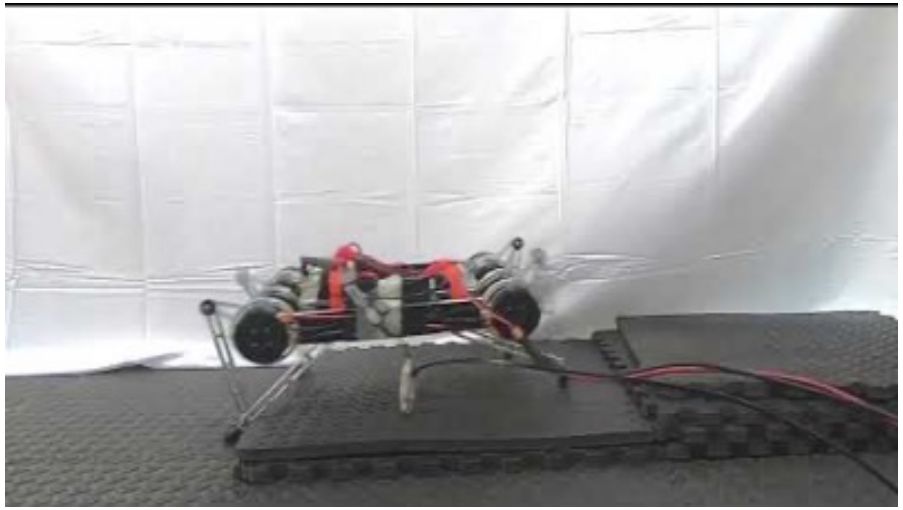
Policy loss, Q – alpha * log pi

Results: Soft Actor-Critic



Haarnoja et al. **Soft Actor-Critic: Algorithms and Applications.** arXiv 2018.

Results: Soft Actor-Critic



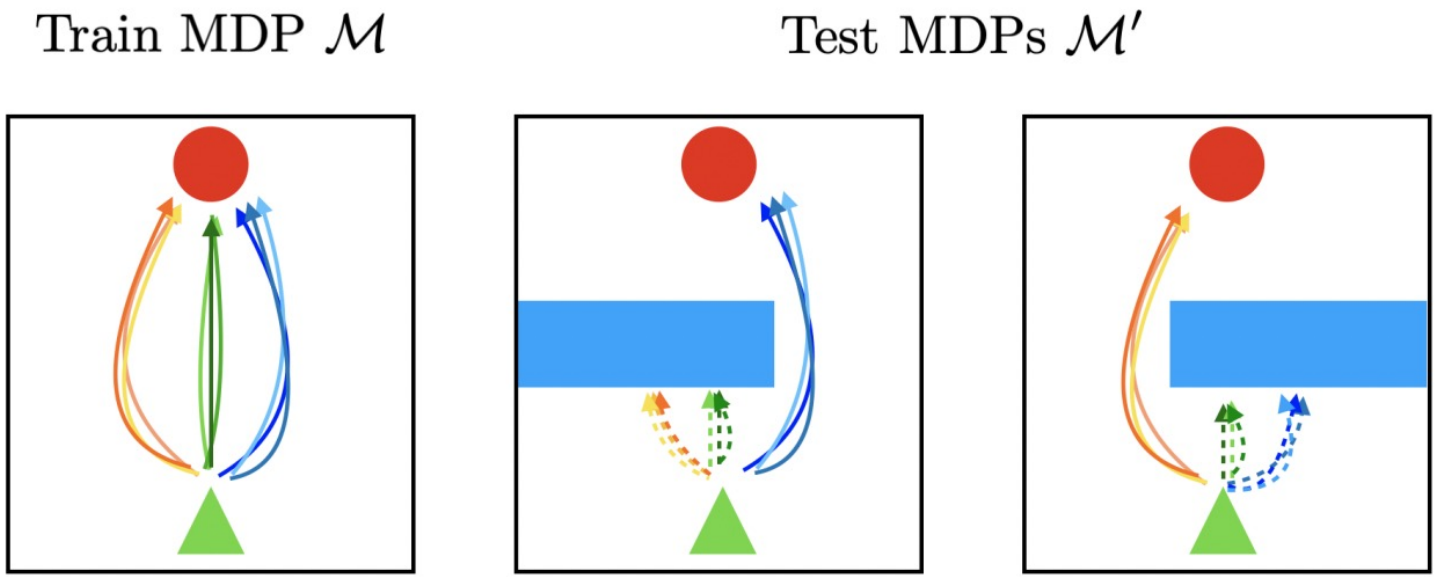
Haarnoja et al. **Soft Actor-Critic: Algorithms and Applications.** arXiv 2018.

Results: Soft Actor-Critic

Smith et al. **Learning to Walk in 20 Minutes With Model-Free Reinforcement Learning**. arXiv 2022

Intuition: When/Why is MaxEnt RL Helpful?

- **More ways to succeed when entropy is backed up in a way**
  - Gives the policy some understanding of how much noise tolerance there is in the future
  - A form of learning a set of **structured exploration** strategies



Kumar et al. **One Solution is Not All You Need: Few-Shot Extrapolation via Structured MaxEnt RL**. NeurIPS 2020.

Intuition: When/Why is MaxEnt RL Helpful?

- **Formal connection to robust RL**
  - Say if the reward function were adversarially chosen
  - How can we then optimize the policy still?
  - Well, the *minimax* optimal solution is to maximize entropy

Theorem 4.1. *Let dynamics $p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})$, policy $\pi(\mathbf{a_t} \mid \mathbf{s_t})$, and reward function $r(\mathbf{s_t}, \mathbf{a_t})$ be given. Assume that the reward function is finite and that the policy has support everywhere (i.e., $\pi(\mathbf{a_t} \mid \mathbf{s_t}) > 0$ for all states and actions). Then there exists a positive constant $\epsilon > 0$ such that the MaxEnt RL objective J_{MaxEnt} is equivalent to the robust RL objective defined by the robust set $\tilde{\mathcal{R}}(\pi)$:*

$$\min_{\tilde{r} \in \tilde{\mathcal{R}}(\pi)} \mathbb{E}\Big[\sum_t \tilde{r}(\mathbf{s_t}, \mathbf{a_t})\Big] = J_{MaxEnt}(\pi; p, r) \quad \forall \pi,$$

where the adversary chooses a reward function from the set

$$\tilde{\mathcal{R}}(\pi) \triangleq \left\{ \tilde{r}(\mathbf{s_t}, \mathbf{a_t}) \;\middle|\; \mathbb{E}_\pi\Big[\sum_t \log \int_{\mathcal{A}} \exp(r(\mathbf{s_t}, \mathbf{a_t}') - \tilde{r}(\mathbf{s_t}, \mathbf{a_t}'))d\mathbf{a_t}'\Big] \leq \epsilon \right\}. \tag{2}$$

Eysenbach and Levine. **Maximum Entropy RL (Provably) Solves Some Robust RL Problems.** ICLR 2022

Intuition: When/Why is MaxEnt RL Helpful?

- **Most important reason: <u>smoothens the optimization landscape</u>**
  - Exponential rate of convergence vs Linear (at best, with TRPO-style)

With Lemmas 14 to 16, we show a $O(e^{-t})$ rate for entropy regularized policy gradient in general MDPs:

Theorem 6. *Suppose $\mu(s) > 0$ for all state s. Using Algorithm 1 with the entropy regularized objective and softmax parametrization and $\eta = (1-\gamma)^3/(8+\tau(4+8\log A))$, there exists a constant $C > 0$ such that for all $t \geq 1$,*

$$\tilde{V}^{\pi_\tau^*}(\rho) - \tilde{V}^{\pi_{\theta_t}}(\rho) \leq \left\| \frac{1}{\mu} \right\|_\infty \cdot \frac{1+\tau\log A}{(1-\gamma)^2} \cdot e^{-C(t-1)}.$$

MaxEnt RL policy gradient

Standard policy gradient

$$V^{(T)}(\rho) \geq V^*(\rho) - \frac{\log|\mathcal{A}|}{\eta T} - \frac{1}{(1-\gamma)^2 T}.$$

Agarwal et al. **On the Theory of Policy Gradient Methods.** JMLR 2021.
Mei et al. **On the Global Convergence Rates of Softmax Policy Gradient Methods**. ICML 2022.