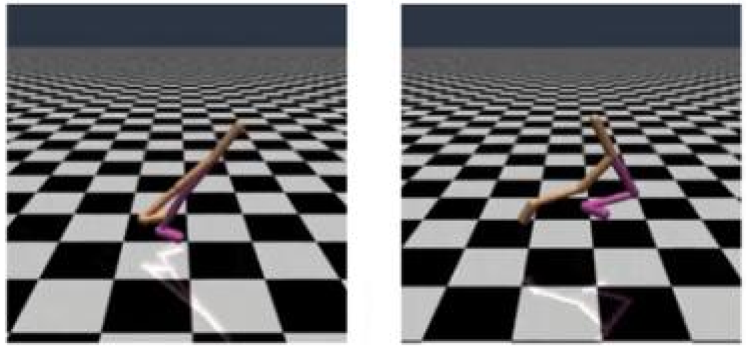
ICML'17

Soft Q-Learning（SQL）

Reinforcement Learning with Deep Energy-Based Policies

Tuomas Haarnoja *[1] Haoran Tang *[2] Pieter Abbeel [1 3 4] Sergey Levine [1]



π_{θ_1} π_{θ_2}

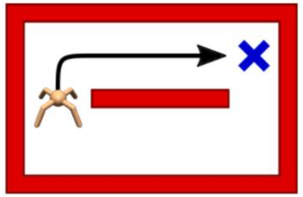
$$\boxed{\pi^* = \underset{\pi}{\operatorname{argmax}} \mathbb{E}_\pi \left[\sum_{t=0}^{T} r_t \right]}$$

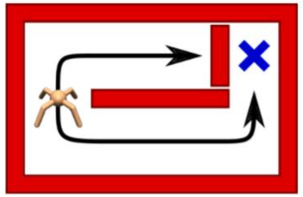
Exploration Problem.

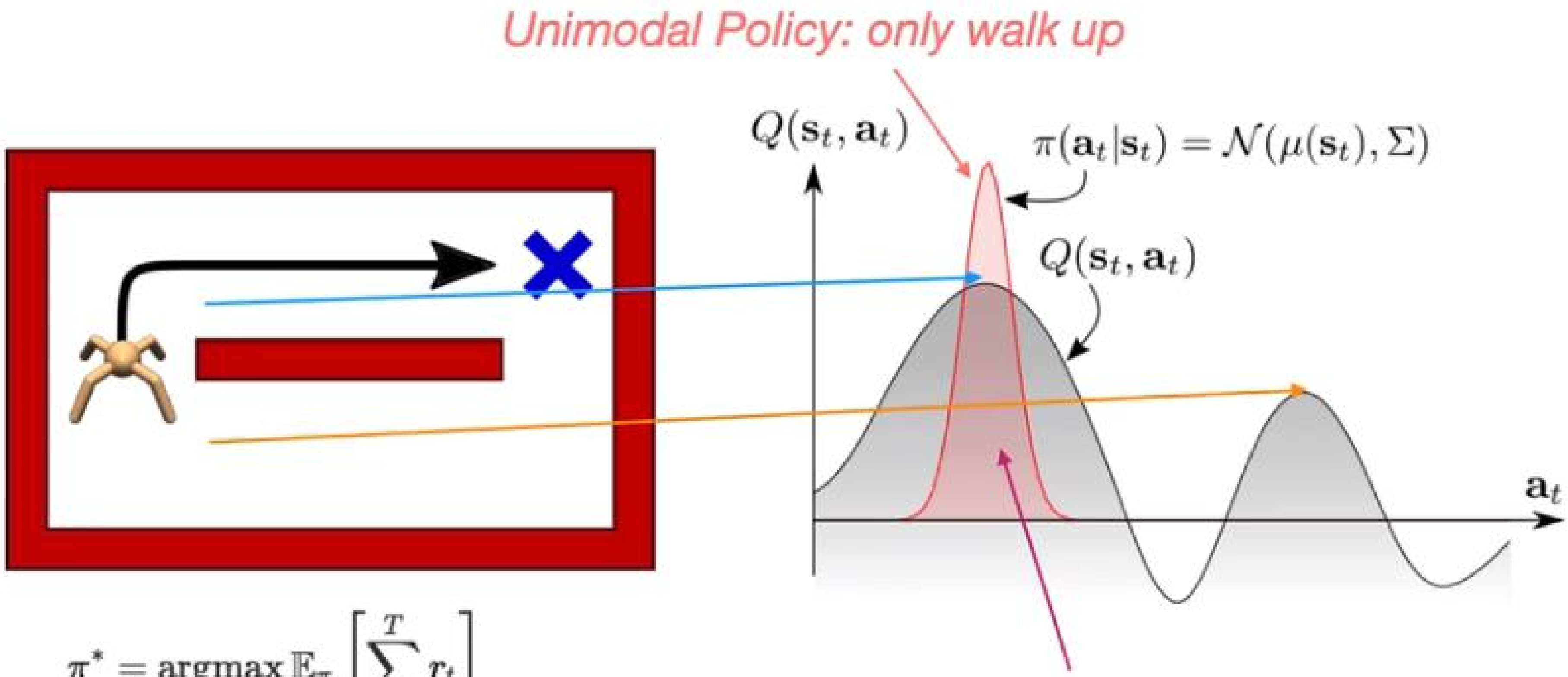
[Motivation]

Standard RL aims to master a single way to solve a given task.

Problem of Standard RL







$$\pi^* = \underset{\pi}{\operatorname{argmax}} \mathbb{E}_\pi \left[\sum_{t=0}^{T} r_t \right]$$

Gaussian policy

$$\pi(\mathbf{a}_t|\mathbf{s}_t) = \mathcal{N}(\mu(\mathbf{s}_t), \Sigma)$$

- Standard RL objective finds a unimodal policy
  - center at the maximal Q-value.
  - add noise for exploration.

Energy-based Policy

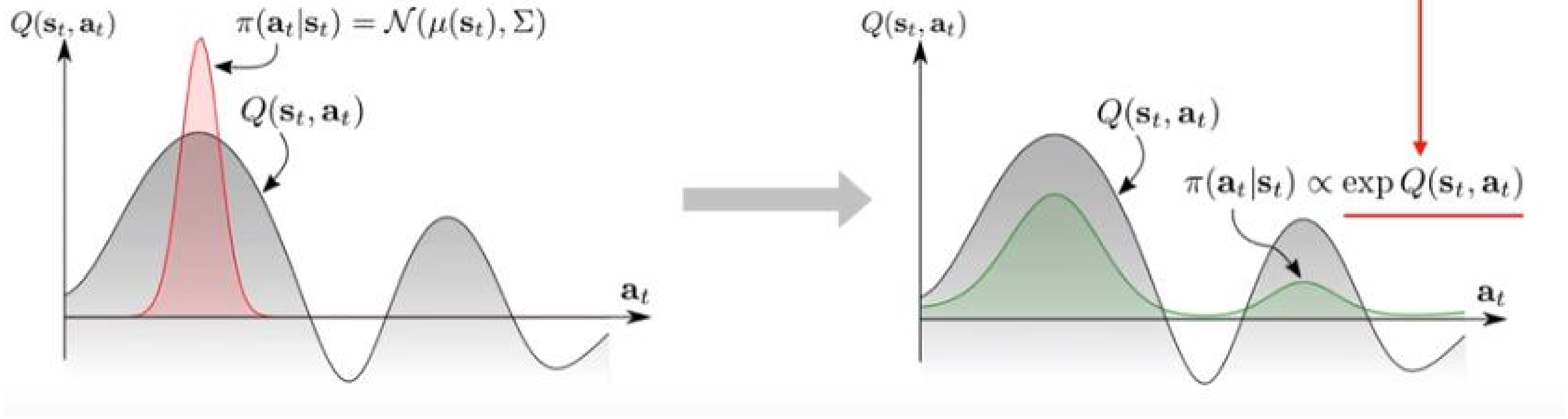
[Intuition] Exploring all promising states while prioritizing the best one.

Define the policy with exponentiated Q-values ==> Boltzmann Distribution

$$\pi(\mathbf{a}_t|\mathbf{s}_t) \propto \exp\left(-\mathcal{E}(\mathbf{s}_t, \mathbf{a}_t)\right)$$

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_\pi\left[\sum_{t=0}^{T} r_t\right]$$

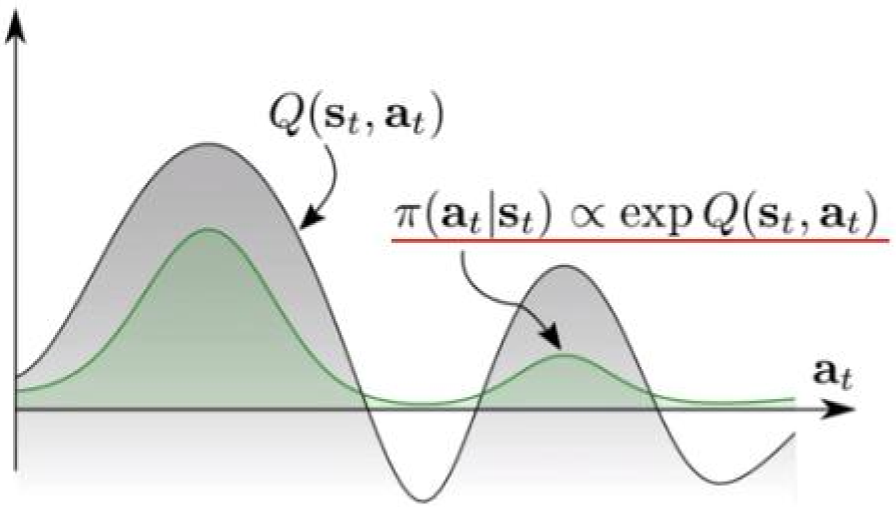
$$\boxed{\mathcal{E}(\mathbf{s}_t, \mathbf{a}_t) = -\frac{1}{\alpha} Q(\mathbf{s}_t, \mathbf{a}_t)}$$



Entropy Regularization

Boltzmann distribution policy:

$$\tilde{\pi}(a_t|s_t) = \frac{\exp(Q^\pi(s_t, a_t))}{\int \exp(Q^\pi(s_t, a))da}$$



At state s_t, the KL divergence:

$$\begin{aligned} D_{KL}\left[\pi(\cdot|s_t) \| \tilde{\pi}(\cdot|s_t)\right] &= \mathbb{E}_{a_t \sim \pi}[\log \pi(\cdot|s_t) - Q^\pi(s_t, a_t) + \log \int \exp(Q^\pi(s_t, a))da] \\ &= -\mathcal{H}(\pi(\cdot|s_t)) - \mathbb{E}_{a_t \sim \pi}[Q^\pi(s_t, a_t)] + \log \int \exp(Q^\pi(s_t, a))da \end{aligned}$$

$$\underbrace{\mathcal{H}(\pi(\cdot|s_t)) + \mathbb{E}_{a_t \sim \pi}[Q^\pi(s_t, a_t)]}_{\text{objective}} = \underline{-D_{KL}\left[\pi(\cdot|s_t) \| \tilde{\pi}(\cdot|s_t)\right]} + \underbrace{\log \int \exp(Q^\pi(s_t, a))da}_{\text{partition function}}$$

Soft Q-Value

Soft Q-value ($\alpha = 1$) is the expectation of the discounted sum of rewards and entropy:

$$Q^\pi(\mathbf{s}, \mathbf{a}) \triangleq r_0 + \mathbb{E}_{\tau\sim\pi, \mathbf{s}_0=\mathbf{s}, \mathbf{a}_0=\mathbf{a}}\left[\sum_{t=1}^{\infty} \gamma^t (r_t + \underline{\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t))})\right]$$ entropy regularized reward

$$\begin{aligned} Q^\pi(\mathbf{s}, \mathbf{a}) &= r(\mathbf{s}, \mathbf{a}) + \gamma\, \mathbb{E}_{\mathbf{s}'\sim p}\left[\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}')) + \mathbb{E}_{\mathbf{a}'\sim\pi(\,\cdot\,|\mathbf{s}')}\left[Q^\pi(\mathbf{s}', \mathbf{a}')\right]\right] \\ &= r(\mathbf{s}, \mathbf{a}) + \gamma\, \mathbb{E}_{\mathbf{s}'\sim p}\left[V^\pi(\mathbf{s}')\right]. \end{aligned}$$

soft-max

hard-max

$$V^\pi(\mathbf{s}) \triangleq \log \int \exp\left(Q^\pi(\mathbf{s}, \mathbf{a})\right)\, d\mathbf{a} \quad \approx \quad V^\pi(s) = \max_a Q^\pi(s, a)$$

$$\pi(a_t|s_t) = \frac{\exp(Q^\pi(s_t, a_t))}{\int \exp(Q^\pi(s_t, a))da} = \exp\left(Q^\pi(\mathbf{s}, \mathbf{a}) - V^\pi(\mathbf{s})\right)$$

soft Bellman operator

$$\mathcal{T}Q(\mathbf{s}, \mathbf{a}) \triangleq r(\mathbf{s}, \mathbf{a}) + \gamma\, \mathbb{E}_{\mathbf{s}'\sim p}\left[\log \int \exp Q(\mathbf{s}', \mathbf{a}')\, d\mathbf{a}'\right]$$

Bellman operator

$$\mathcal{T}Q(s, a) = r(s, a) + \gamma \mathbb{E}_{s'\sim p}\left[\max_{a'} Q(s', a')\right]$$

Soft Q-Learning

Soft Bellman Backup Operator $\mathcal{T}$

Contraction

Theorem 1. *Soft Q-iteration. Let Q and V be bounded and* $\int_{\mathcal{A}} \exp(\frac{1}{\alpha}Q(\mathbf{s}_t, \mathbf{a}'))d\mathbf{a}' < \infty$, $\forall \mathbf{s}_t \in \mathcal{S}$, *and assume that* $Q^* < \infty$ *exists. Then the fixed-point iteration*

$$Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow r(\mathbf{s}_t, \mathbf{a}_t) + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[V(\mathbf{s}_{t+1})\right],\ \forall \mathbf{s}_t, \mathbf{a}_t \tag{3.6}$$

$$V(\mathbf{s}_t) \leftarrow \alpha \log\left(\int_{\mathcal{A}} \exp\left(\frac{1}{\alpha}Q(\mathbf{s}_t, \mathbf{a}')\right) d\mathbf{a}'\right),\ \forall \mathbf{s}_t \tag{3.7}$$

converges to Q^* *and* V^*, *respectively.*

Boltzmann Distribution

Soft max

Entropy regularized reward

Optimal Soft Q-function: $Q^*_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t) = r_t + \mathbb{E}_{(\mathbf{s}_{t+1},\ldots)\sim\rho_\pi}\left[\sum_{l=1}^{\infty} \gamma^l \left(r_{t+l} + \underline{\alpha\mathcal{H}\left(\pi^*_{\mathrm{MaxEnt}}(\cdot\,|\mathbf{s}_{t+l})\right)}\right)\right]$

Optimal Soft value function: $V^*(\mathbf{s}_t) = \underline{\alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha}Q^*(\mathbf{s}_t, \mathbf{a}')\right) d\mathbf{a}'} \longleftrightarrow V^\pi(s) = \max_a Q^\pi(s, a)$

Optimal policy: $\pi^*_{\mathrm{MaxEnt}}(\mathbf{a}_t|\mathbf{s}_t) = \exp\left(\frac{1}{\alpha}\left(Q^*_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t) - V^*_{\mathrm{soft}}(\mathbf{s}_t)\right)\right) \longleftrightarrow \pi(a\,|\,s) = \max_a Q(s, a)$

Problems of Soft Q-learning

Soft Bellman Backup Operator

$$\boxed{\begin{aligned} Q_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) &\leftarrow r_t + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1}\sim p_{\mathbf{s}}}\left[V_{\text{soft}}(\mathbf{s}_{t+1})\right], \ \forall \mathbf{s}_t, \mathbf{a}_t \\ V_{\text{soft}}(\mathbf{s}_t) &\leftarrow \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q_{\text{soft}}(\mathbf{s}_t, \mathbf{a}')\right) d\mathbf{a}', \ \forall \mathbf{s}_t \end{aligned}}$$

$$\pi(a_t|s_t) = \frac{\exp(Q^\pi(s_t, a_t))}{\int \exp(Q^\pi(s_t, a))da}$$

Hard to sample from π

Integral over infinite states and actions

$$\underbrace{V^*(\mathbf{s}_t) = \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q^*(\mathbf{s}_t, \mathbf{a}')\right) d\mathbf{a}'}_{\text{intractable}} \xrightarrow{\text{IS}} V_\theta(\mathbf{s}_t) = \alpha \log \mathbb{E}_{q_{\mathbf{a}'}}\left[\frac{\exp\left(\frac{1}{\alpha} Q_\theta(\mathbf{s}_t, \mathbf{a}')\right)}{q_{\mathbf{a}'}(\mathbf{a}')}\right]$$

arbitrary distribution

Minimize soft Bellman error $|\mathcal{T}Q - Q|$ ==> sample (s_t, a_t, s_{t+1}) from ER and SGD

$$J_Q(\theta) = \underbrace{\mathbb{E}_{\mathbf{s}_t\sim q_{\mathbf{s}}, \mathbf{a}_t\sim q_{\mathbf{a}}}}_{\text{arbitrary sampling distribution}}\left[\frac{1}{2}\left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[\underbrace{V_{\bar\theta}(\mathbf{s}_{t+1})}_{\text{target value network}}\right]\right)\right)^2\right]$$

Approximate Sampling and Stein Variational Gradient Descent (SVGD)

$$\pi(a_t|s_t) = \frac{\exp(Q^\pi(s_t, a_t))}{\int \exp(Q^\pi(s_t, a))da}$$

A stochastic NN $\mathbf{a}_t = f^\phi(\xi; \mathbf{s}_t)$ maps noise ξ to action samples.

Stochastic Sampling Network

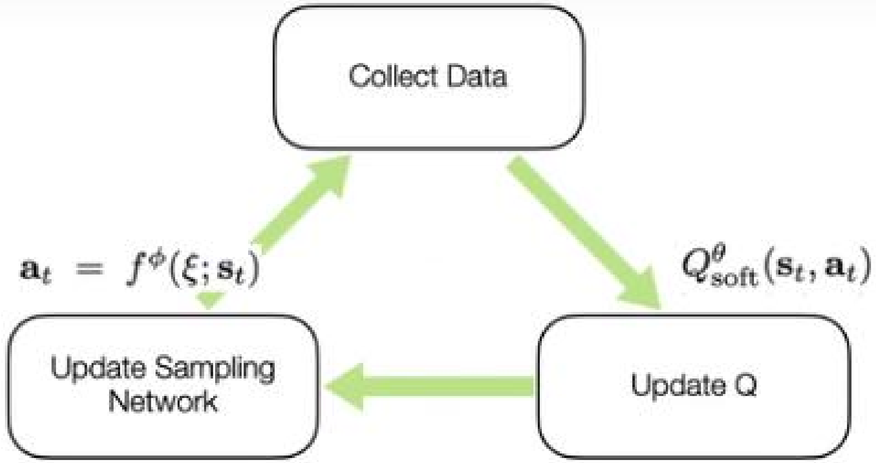
Objective $J_\pi(\phi; \mathbf{s}_t) = \mathrm{D}_{\mathrm{KL}}\left(\pi_\phi(\,\cdot\,|\mathbf{s}_t) \,\middle\|\, \exp\left(\frac{1}{\alpha}\left(Q_\theta(\mathbf{s}_t, \cdot\,) - V_\theta(\mathbf{s}_t)\right)\right)\right)$

Suppose we have a set of independent, state conditioned action samples $\{\mathbf{a}_t^{(i)} = f_\phi(\xi^{(i)}; \mathbf{s}_t)\}$, given by the sampling network, and we "perturb" them in some directions $\Delta f_\phi(\xi^{(i)}; \mathbf{s}_t)$, then the KL divergence induced by the samples can be reduced. Stein variational gradient descent (Liu & Wang, 2016) provides the most greedy directions as a functional

$$\Delta f_\phi(\,\cdot\,; \mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi_\phi}\left[\kappa(\mathbf{a}_t, f_\phi(\,\cdot\,; \mathbf{s}_t)) \nabla_{\mathbf{a}'} Q_\theta(\mathbf{s}_t, \mathbf{a}')\big|_{\mathbf{a}'=\mathbf{a}_t} + \alpha\, \nabla_{\mathbf{a}'} \kappa(\mathbf{a}', f_\phi(\,\cdot\,; \mathbf{s}_t))\big|_{\mathbf{a}'=\mathbf{a}_t}\right], \quad (3.11)$$

$$\frac{\partial J_\pi}{\partial \mathbf{a}_t} \propto \Delta f^\phi \longrightarrow \frac{\partial J_\pi(\phi; \mathbf{s}_t)}{\partial \phi} \propto \mathbb{E}_\xi\left[\Delta f^\phi(\xi; \mathbf{s}_t) \frac{\partial f^\phi(\xi; \mathbf{s}_t)}{\partial \phi}\right]$$

Wang, D. and Liu, Q. Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv preprint arXiv:1611.01722*, 2016.



$$V_\theta(\mathbf{s}_t) = \alpha \log \mathbb{E}_{q_{\mathbf{a}'}}\left[\frac{\exp\left(\frac{1}{\alpha} Q_\theta(\mathbf{s}_t, \mathbf{a}')\right)}{q_{\mathbf{a}'}(\mathbf{a}')}\right]$$

$$J_Q(\theta) = \mathbb{E}_{\mathbf{s}_t \sim q_{\mathbf{s}}, \mathbf{a}_t \sim q_{\mathbf{a}}}\left[\frac{1}{2}\left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[V_{\bar\theta}(\mathbf{s}_{t+1})\right]\right)\right)^2\right]$$

Stein Variational Gradient Descent

$$\Delta f^{\phi}(\,\cdot\,; \mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi^{\phi}}\Big[\kappa(\mathbf{a}_t, f^{\phi}(\,\cdot\,; \mathbf{s}_t)) \nabla_{\mathbf{a}'} Q^{\theta}_{\text{soft}}(\mathbf{s}_t, \mathbf{a}')\big|_{\mathbf{a}'=\mathbf{a}_t} + \alpha\, \nabla_{\mathbf{a}'} \kappa(\mathbf{a}', f^{\phi}(\,\cdot\,; \mathbf{s}_t))\big|_{\mathbf{a}'=\mathbf{a}_t}\Big], \tag{13}$$

$$\frac{\partial J_\pi(\phi; \mathbf{s}_t)}{\partial \phi} \propto \mathbb{E}_{\xi}\left[\Delta f^{\phi}(\xi; \mathbf{s}_t) \frac{\partial f^{\phi}(\xi; \mathbf{s}_t)}{\partial \phi}\right]$$

Algorithm 1 Soft Q-learning

$\theta, \phi \sim$ some initialization distributions.
Assign target parameters: $\bar\theta \leftarrow \theta$, $\bar\phi \leftarrow \phi$.
$\mathcal{D} \leftarrow$ empty replay memory.
for each epoch **do**
  for each t **do**
    Collect experience
      Sample an action for $\mathbf{s}_t$ using f^{ϕ}:
      $\mathbf{a}_t \leftarrow f^{\phi}(\xi; \mathbf{s}_t)$ where $\xi \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.
      Sample next state from the environment:
      $\mathbf{s}_{t+1} \sim p_{\mathbf{s}}(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$.
      Save the new experience in the replay memory:
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$.
    Sample a minibatch from the replay memory
      $\{(\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)}, r_t^{(i)}, \mathbf{s}_{t+1}^{(i)})\}_{i=0}^{N} \sim \mathcal{D}$.
    Update the soft Q-function parameters
      Sample $\{\mathbf{a}^{(i,j)}\}_{j=0}^{M} \sim q_{\mathbf{a}'}$ for each $\mathbf{s}_{t+1}^{(i)}$.
      Compute empirical soft values $\hat{V}^{\bar\theta}_{\text{soft}}(\mathbf{s}_{t+1}^{(i)})$ in (10).
      Compute empirical gradient $\hat\nabla_\theta J_Q$ of (11).
      Update θ according to $\hat\nabla_\theta J_Q$ using ADAM.
    Update policy
      Sample $\{\xi^{(i,j)}\}_{j=0}^{M} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ for each $\mathbf{s}_t^{(i)}$.
      Compute actions $\mathbf{a}_t^{(i,j)} = f^{\phi}(\xi^{(i,j)}, \mathbf{s}_t^{(i)})$.
      Compute Δf^{ϕ} using empirical estimate of (13).
      Compute empiricial estimate of (14): $\hat\nabla_\phi J_\pi$.
      Update ϕ according to $\hat\nabla_\phi J_\pi$ using ADAM.
  end for
  if epoch *mod* update_interval $= 0$ **then**
    Update target parameters: $\bar\theta \leftarrow \theta$, $\bar\phi \leftarrow \phi$.
  end if
end for

Experiment

(1) Can SQL capture a multi-modal policy distribution?
(2) Can SQL help exploration?
(3) Can SQL serve as a good initialization for fine-tuning on different tasks.

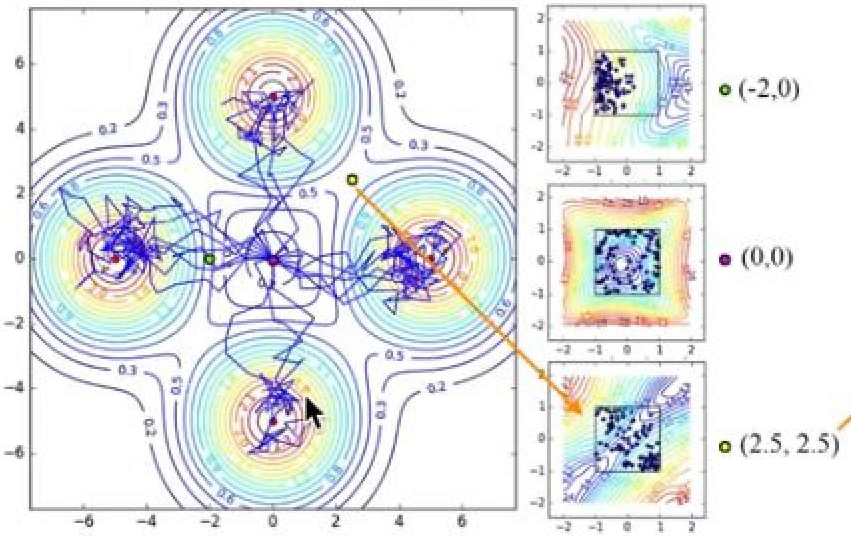
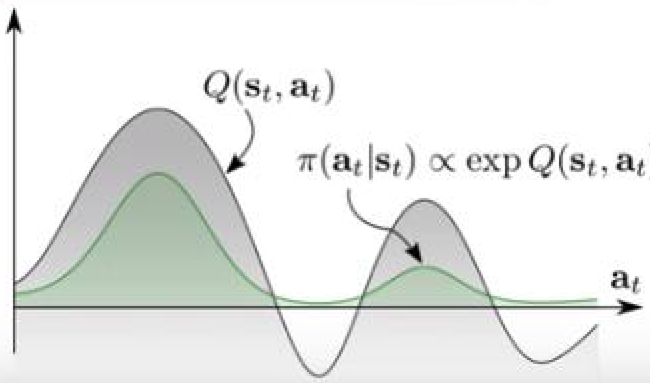
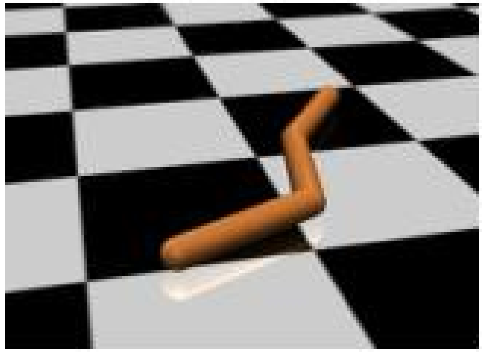


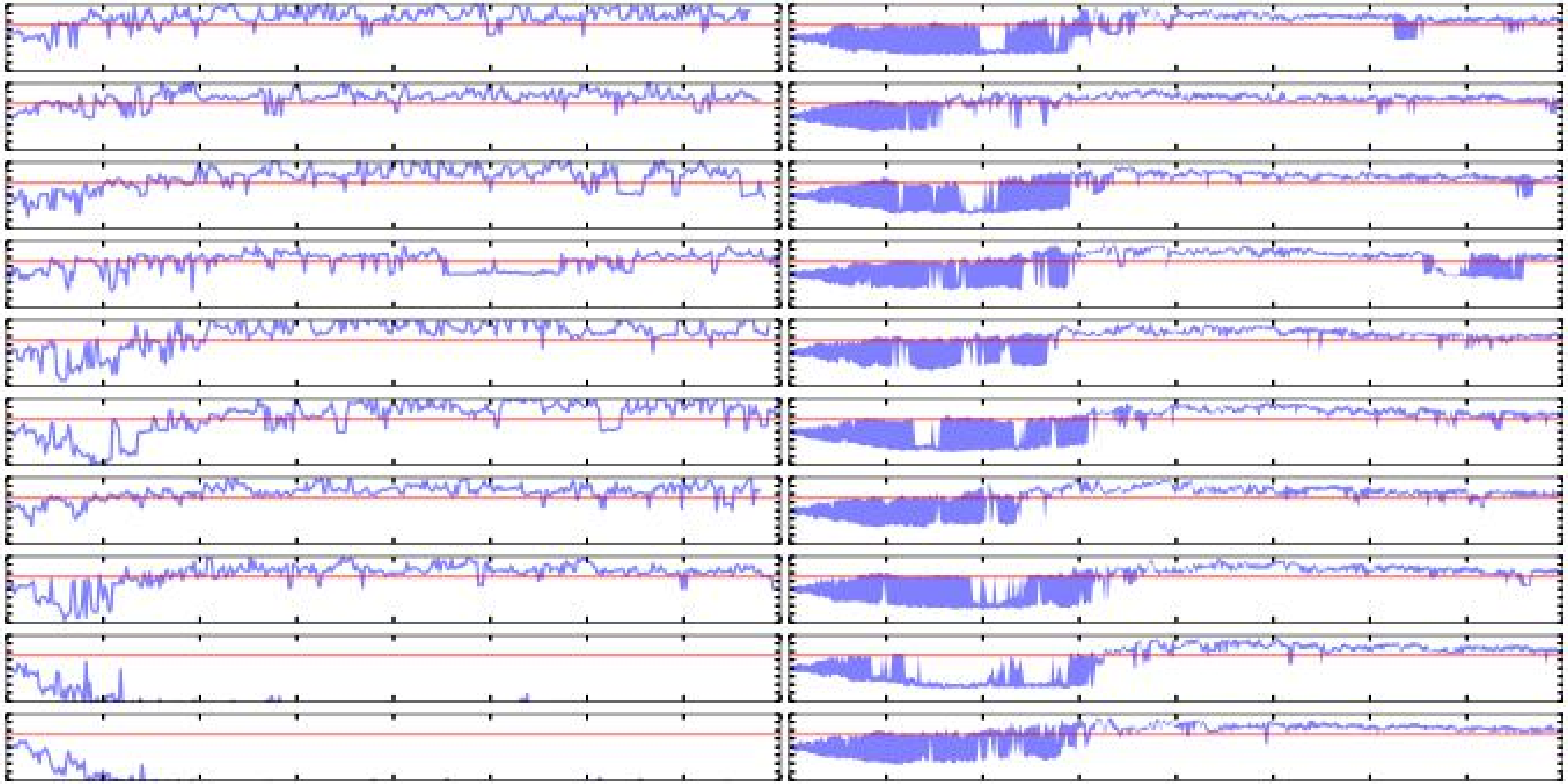
Figure 1. Illustration of the 2D multi-goal environment. Left: trajectories from a policy learned with our method (solid blue lines). The x and y axes correspond to 2D positions (states). The agent is initialized at the origin. The goals are depicted as red dots, and the level curves show the reward. Right: Q-values at three selected states, depicted by level curves (red: high values, blue: low values). The x and y axes correspond to 2D velocity (actions) bounded between -1 and 1. Actions sampled from the policy are shown as blue stars. Note that, in regions (e.g. (2.5, 2.5)) between the goals, the method chooses multimodal actions.



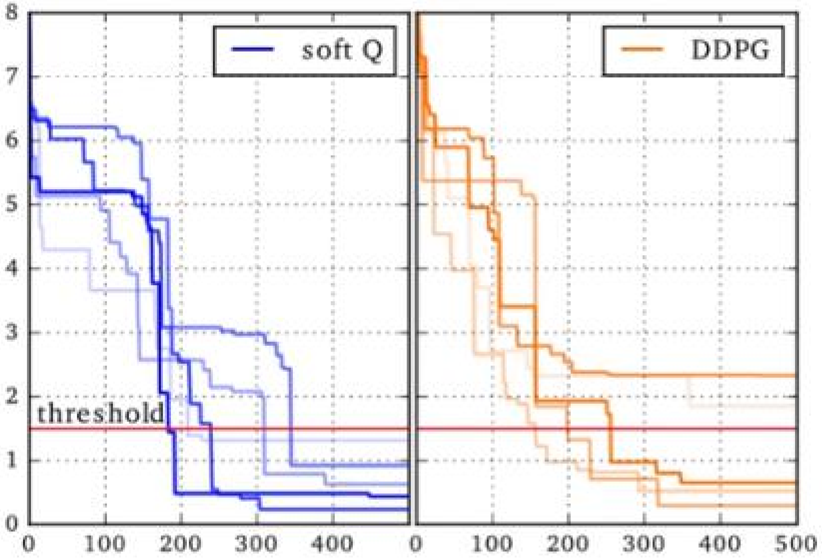
for success. The first experiment uses a simulated swimming snake (see Figure 2), which receives a reward equal to its speed along the x-axis, either forward or backward. However, once the swimmer swims far enough forward, it crosses a "finish line" and receives a larger reward. Therefore, the best learning strategy is to explore in both directions until the bonus reward is discovered, and then commit to swimming forward. As illustrated in Figure 6 in



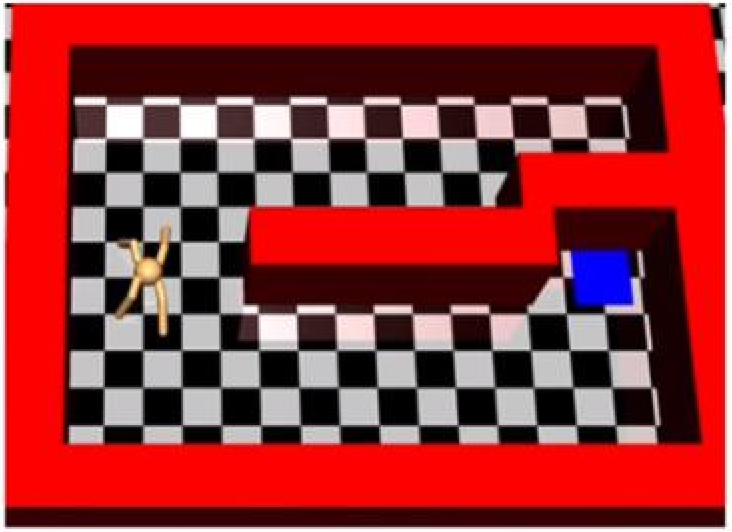
(a) Swimming snake



(a) DDPG (b) Soft Q-learning

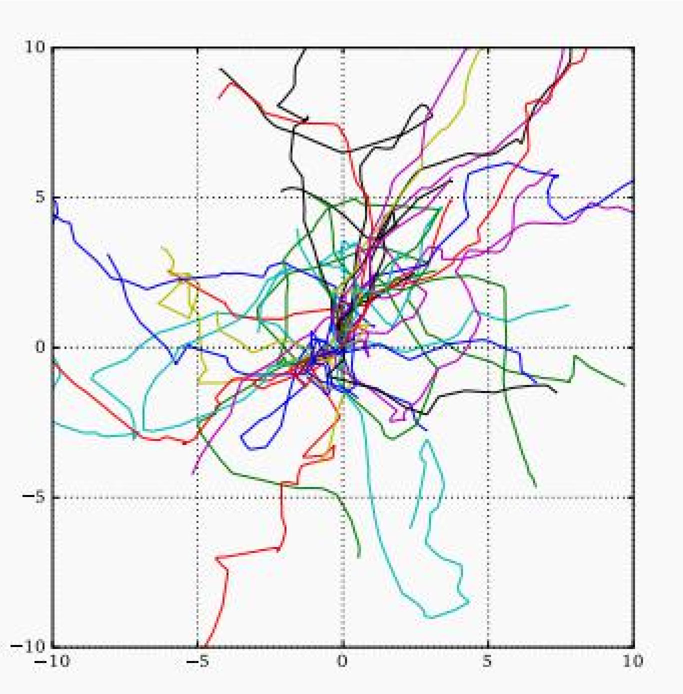


(b) Quadruped (lower is better)



(b) Quadrupedal robot

the performance of DDPG and our method. The curves show the minimum distance to the target achieved so far and the threshold equals the minimum possible distance if the robot chooses the upper passage. Therefore, successful exploration means reaching below the threshold. All policies trained with our method manage to succeed, while only 60% policies trained with DDPG converge to choosing the lower passage.



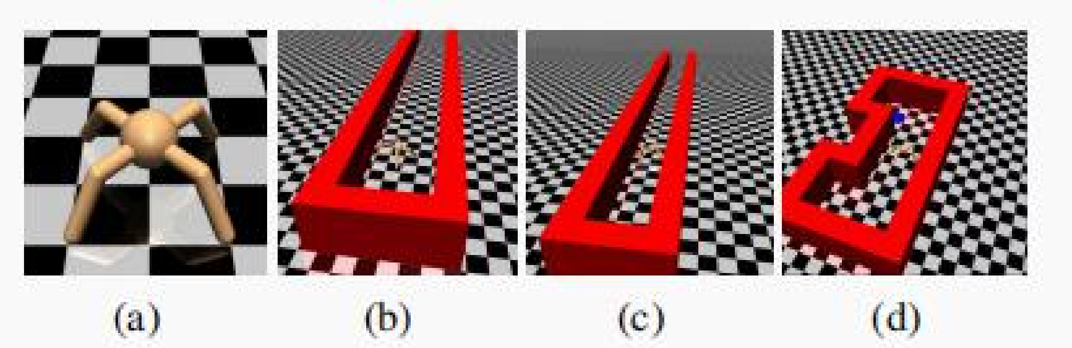


Figure 4. Quadrupedal robot (a) was trained to walk in random directions in an empty pretraining environment (details in Figure 7, see Appendix D.3), and then finetuned on a variety of tasks, including a wide (b), narrow (c), and U-shaped hallway (d).

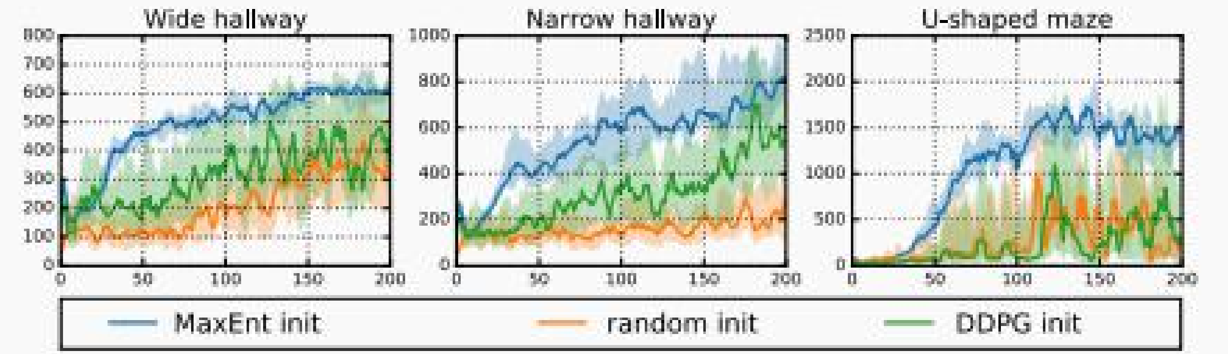


Figure 5. Performance in the downstream task with fine-tuning (MaxEnt) or training from scratch (DDPG). The x-axis shows the training iterations. The y-axis shows the average discounted return. Solid lines are average values over 10 random seeds. Shaded regions correspond to one standard deviation.