Deep Reinforcement Learning

Lecture 7 — Handling Continuous Action Space

國立清華大學
NATIONAL TSING HUA UNIVERSITY

National Tsing Hua University
Department of Computer Science

Outline

- Continuous Action Space and MuJoCo
- DPG and DDPG
- TD3
- HER
- Soft Q-learning, SAC

Continuous Action Space

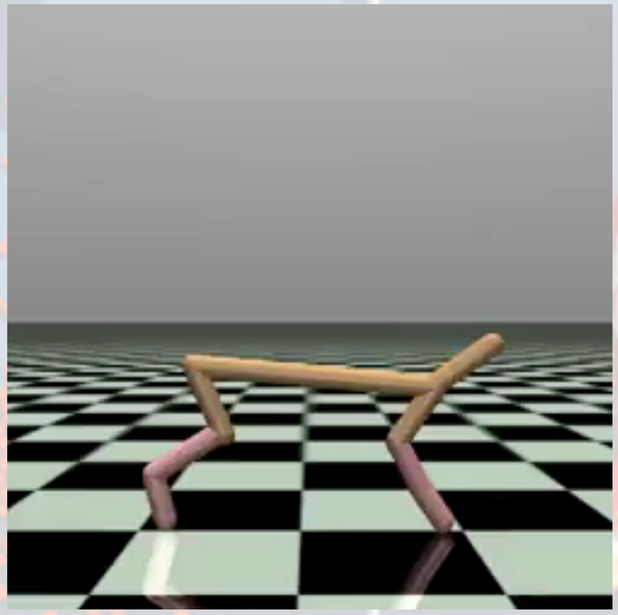
Examples

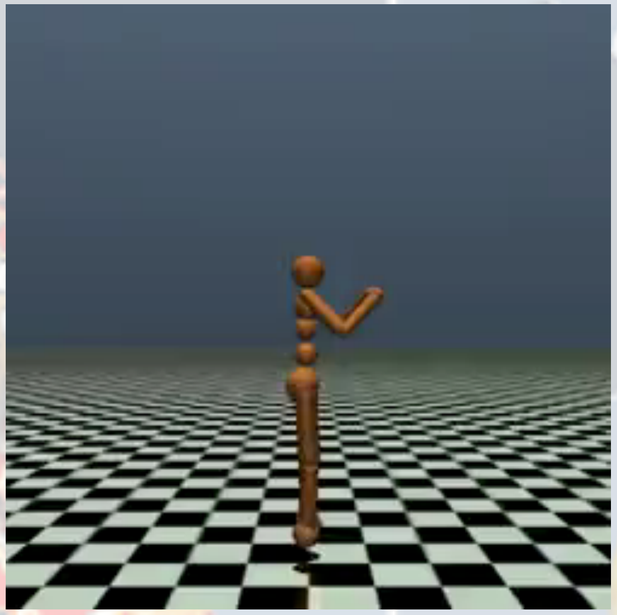
- In many cases, the action space can be continuous.
  - Car driving
  - Drones
  - Robotic Arms
  - AGVs
  - …, and etc.

Mujoco Environments

Mujoco — **Mu**lti-**Jo**int dynamics with **Co**ntact

- A physics engine aiming to facilitate research and development in robotics, biomechanics, graphics and animation.
  - **Demo video**: tinyurl.com/3shwyyjf
  - **Web**: http://www.mujoco.org/

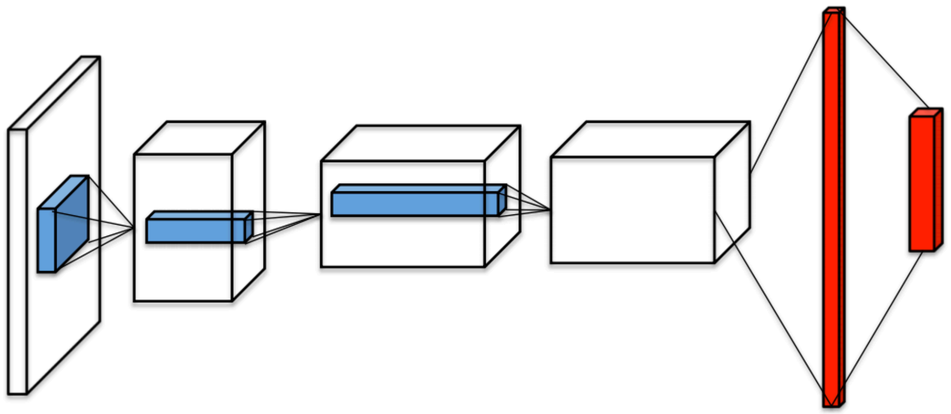




Continuous Action Space

Recap of the discrete action space

- Example of the RL methods for Discrete Action Space
  - The DQN family
  - The actions are countable in general



How to output continuous value?

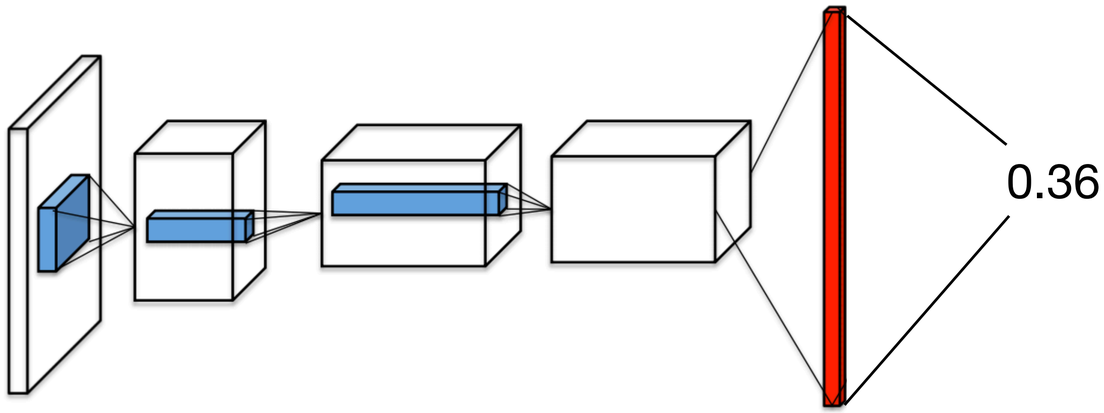
Continuous Action Space

Approaches for dealing with continuous action space

- Quantization of the output values

$$[-1,1] \rightarrow -1, -0.8, -0.6, \ldots 0.8, 1$$

- Directed output continuous values



Limitations of DQN

The infeasibility in continuous and high-dimensional action space

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma \max_{a} Q(s',a) - Q(s,a)]$$

This term tries to find the maximum Q value of the next states.

However, this is **infeasible** when **the action space is continuous or high dimensional**.

How do we use a deep neural network to predict an action,

which can make the return maximized?

Deterministic Policy Gradient (DPG)

The role of the actor

The mission of the actor is to

find an action which leads to the maximized Q

Actor
(A deep neural network)

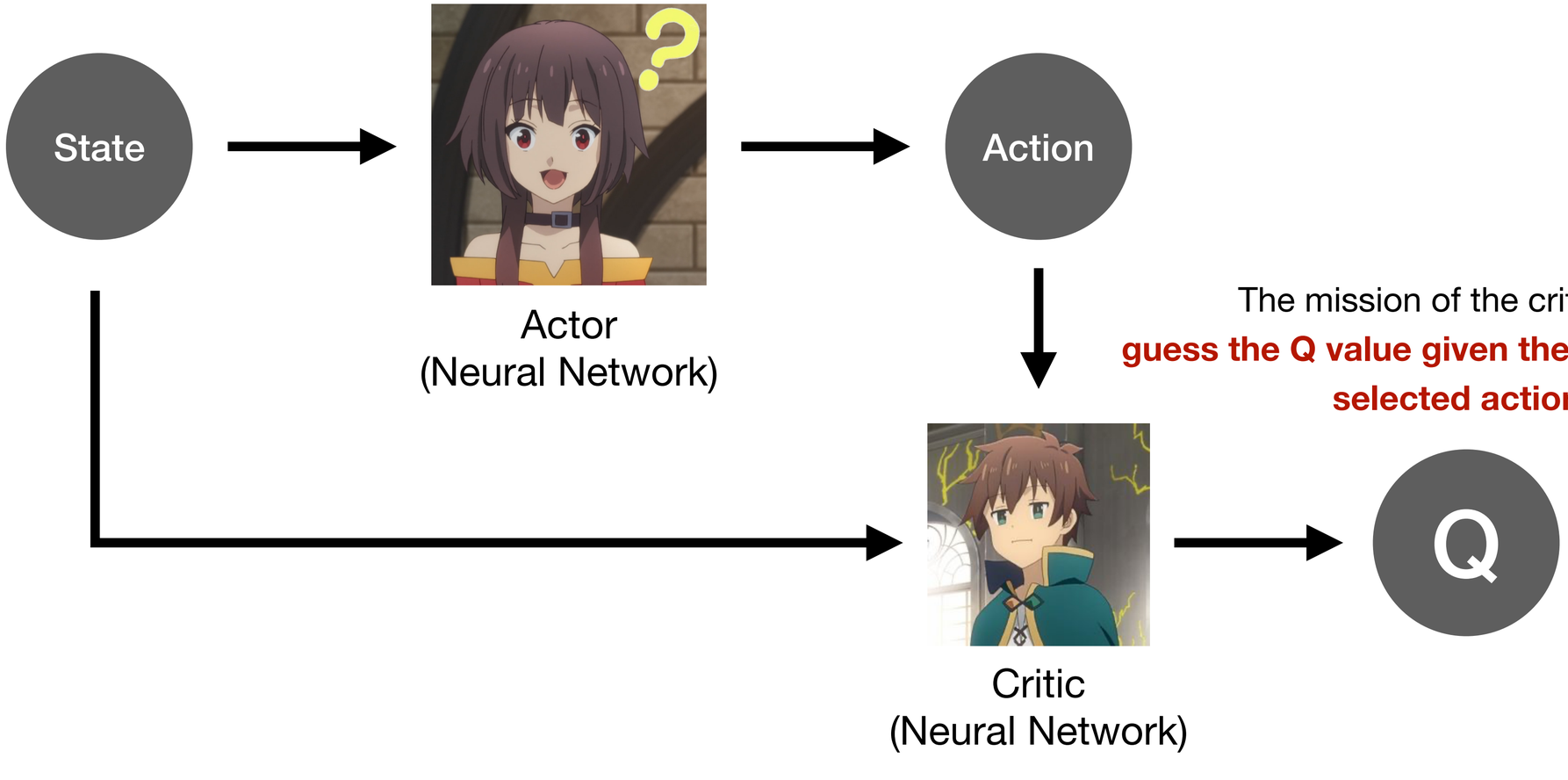
We also **need a critic to guess the outcome of the selected action**

Deterministic Policy Gradient

The role of the actor and the critic

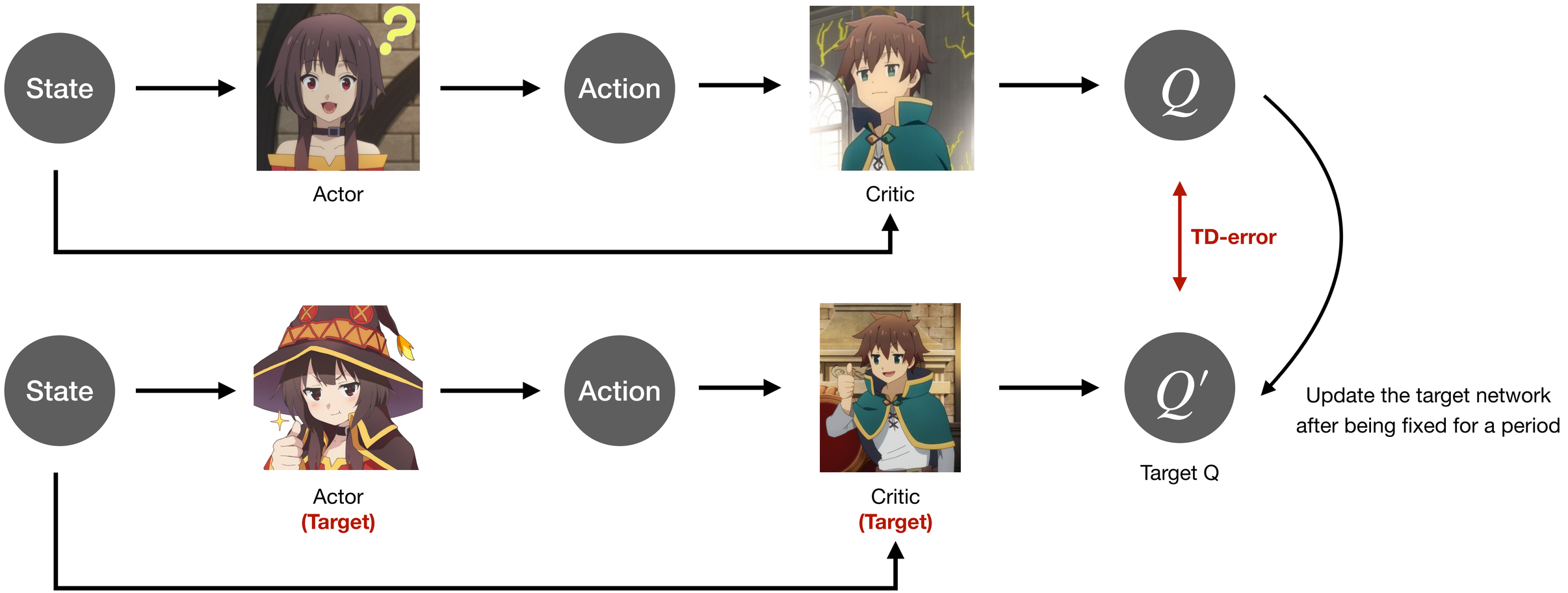
The mission of the actor is to

find an action which leads to the maximized Q



Deterministic Policy Gradient

The update procedure for the actor and the critic (similar to DQN)



Deep Deterministic Policy Gradient (DDPG)

The main concepts of DDPG

- An actor-critic method developed for dealing with continuous action space problems
- DDPG updates the critic Q in the same way as DQN
- DDPG updates the actor μ via **Deterministic Policy Gradient**
- The agent takes an action according to $a = \mu(s, \theta_\mu) + \mathcal{N}_{noise}$

Deep Deterministic Policy Gradient (DDPG)

A comparison of the stochastic and deterministic policy gradients

- Deterministic Policy Gradient

$$\nabla_\theta J(\theta) = \sum_{s \in S} d_\pi(s) \nabla_\theta \pi_\theta(s) \nabla_a Q_\pi(s, a) \,|_{a=\pi_\theta(s)}$$

- Stochastic Policy Gradient

$$\nabla_\theta J(\theta) = \sum_{s \in S} d_\pi(s) \sum_a \nabla_\theta \pi_\theta(a \,|\, s) Q_\pi(s, a)$$

Deep Deterministic Policy Gradient (DDPG)

Formulation of the policy gradient

- The formulation of the policy gradient in a continuous action space

$$\nabla_\theta J(\theta) = \sum_{s \in S} d_\pi(s) \nabla_\theta \pi_\theta(s) \nabla_a Q_\pi(s, a) \mid_{a=\pi_\theta(s)}$$

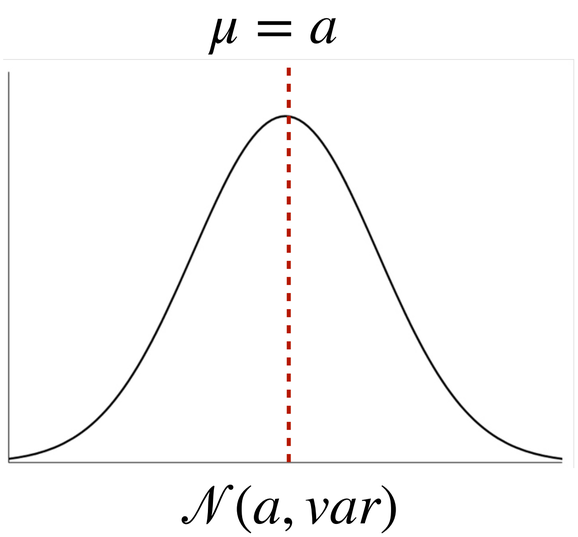
- No summation over the action space.
- It is more easily to train an agent based on the above formulation in a high dimensional action space.

Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient

- However, deterministic policy would lack exploratory behaviors
- Solution: adding noise terms during the training phase

$$a = \mu(s, \theta_\mu) + \mathcal{N}_{noise}$$



  - $\mathcal{N}_{noise}$: Ornstein-Uhlenbeck process (Link)
  - $\mu(s, \theta_\mu)$: The policy function parameterized by θ_μ
  - Adjust the variance of $\mathcal{N}_{noise}$ to control the extent of exploration behavior

Deep Deterministic Policy Gradient (DDPG)

Pseudo Code (1/2)

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R

Deep Deterministic Policy Gradient (DDPG)

Pseudo Code (2/2)

Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

end for
end for

TD3

Twin Delayed Deep Deterministic Policy Gradient

- An **actor-critic** based method
- TD3 is proposed to mitigate the overestimation problem in DDPG
- The contributions introduced in TD3:
    - Clipped double Q-learning
    - **Delayed updates** of the policy and the target network
    - Target policy **smooth regularization**

TD3

Recap: Overestimation bias in double Q-learning

- Double Q-learning
  - $y_A = r_t + \gamma Q_{\theta_A}(s_{t+1}, \arg\max_{a'} Q_{\theta_B}(s_{t+1}, a'))$
  - $y_B = r_t + \gamma Q_{\theta_B}(s_{t+1}, \arg\max_{a'} Q_{\theta_A}(s_{t+1}, a'))$
- **Double Q-learning** uses Q-functions (i.e., Q_{θ_A} and Q_{θ_B} in the above equation), where each network selects the action for the other

TD3

Recap: How double-DQN deals with the overestimation problem

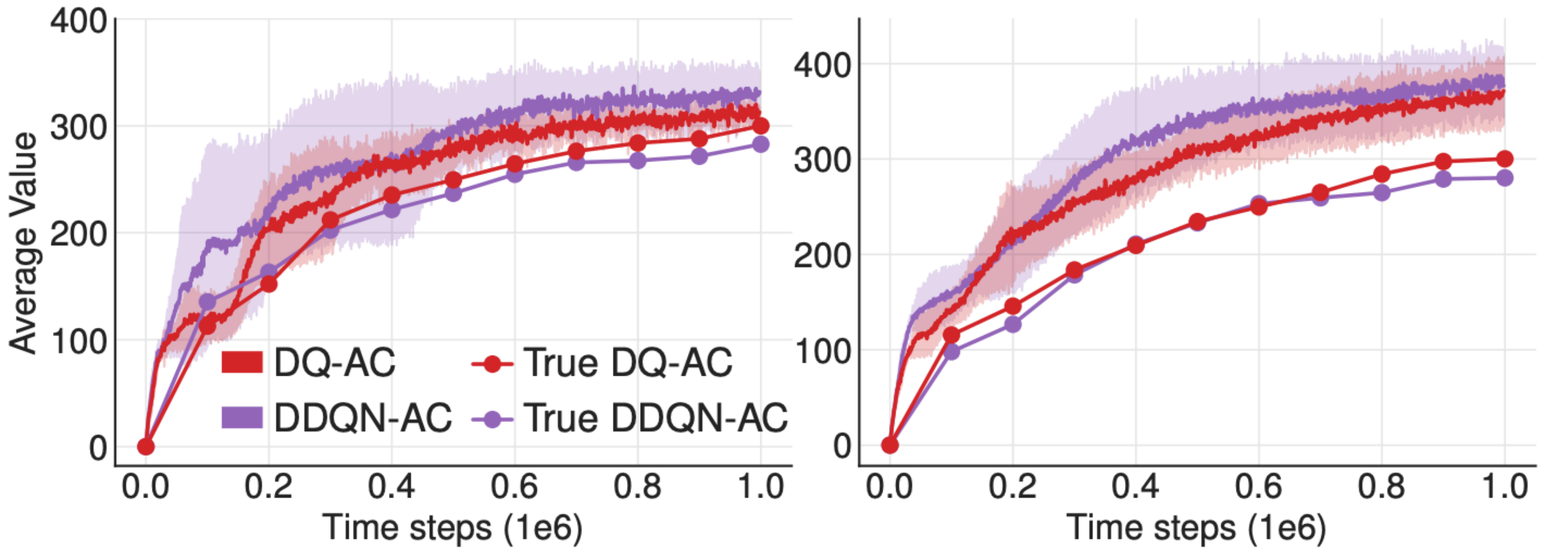
- DQN inherently contains two networks
  - Q_{θ^-} is target network, Q_θ is training network

- The update target of double-DQN is formulated as follows:
  - $y = r_t + \gamma Q_{\theta^-}(s_{t+1}, \arg\max_{a'} Q_\theta(s_{t+1}, a'))$
  - By using one network as the target and another for action selection, double-DQN is able to mitigate the overestimation problem

TD3

Overestimation bias in actor-critic

- Value estimates of the actor critic variants of **double DQN (DDQN-AC)** and **double Q-learning (DQ-AC)**



(a) Hopper-v1 (b) Walker2d-v1

TD3

Clipped double Q-learning for actor-critic

- TD3 introduces the concept of **clipped critic**
  - It replaces the original formulation of the critic
  - It is applicable to any actor-critic method
- TD3 uses two critics $Q_{\theta_1}, Q_{\theta_2}$ for estimating the Q-values, where the action is selected by the actor π_ϕ
  - It then chooses the minimum value between the two estimates as the target value

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i}(s_{t+1}, \pi_\phi(s_{t+1}))$$

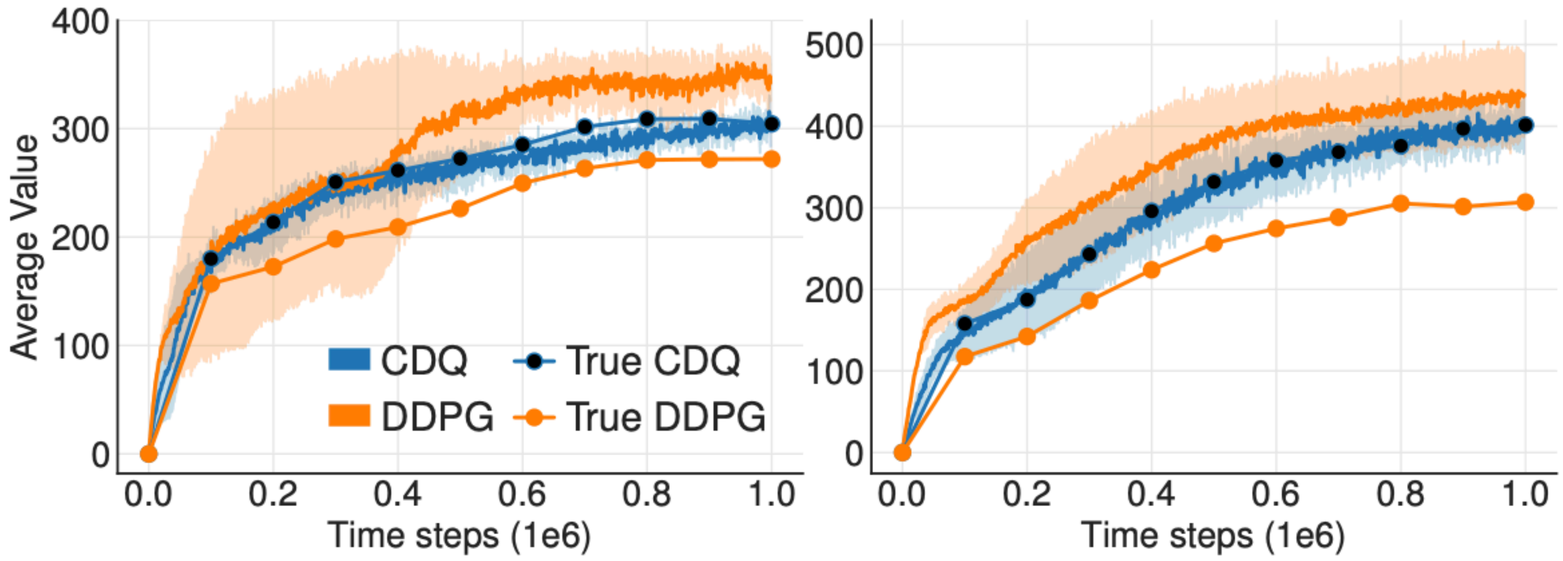
clipped

Same actor

TD3

Measuring overestimation bias

- Clipped double Q-learning for actor-critic v.s. the normal one
  - The results shows that the estimated values are very close to the true ones

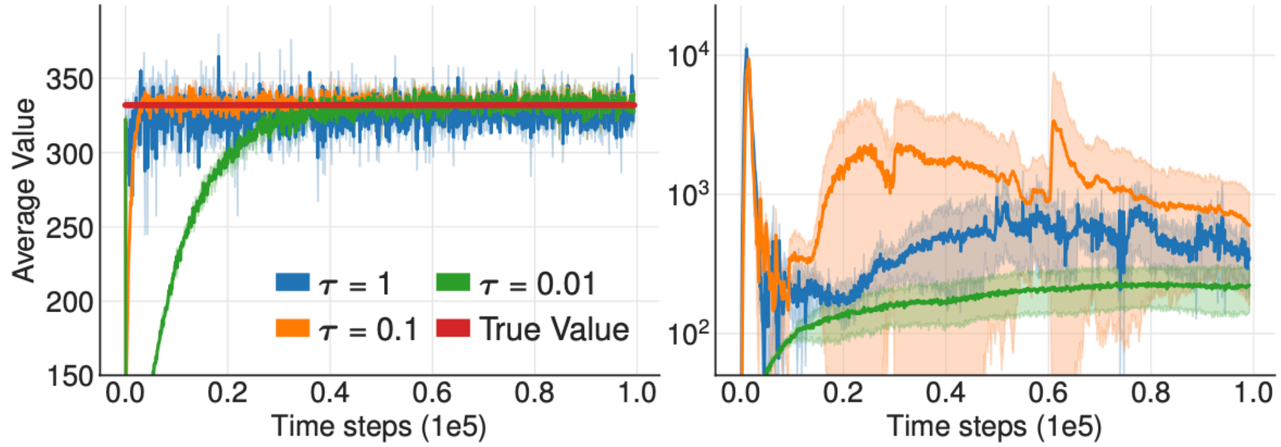


(a) Hopper-v1 (b) Walker2d-v1

TD3

Case study: different update rates for the policy and the target network

- The policy should be updated in a slower, less frequent rate
  - τ represents the update ratio of the target network (refer to p. 27)
  - The larger τ is, the faster the target network is updated
- If the policy is fixed, different values of τ can all help the Q-function to converge (although with slightly different speeds)
- However, if the policy is also learning, a larger τ may cause the learning Q-function to **fluctuate drastically**
  - This observation implies that the policy and the target network should be updated in a slower rate

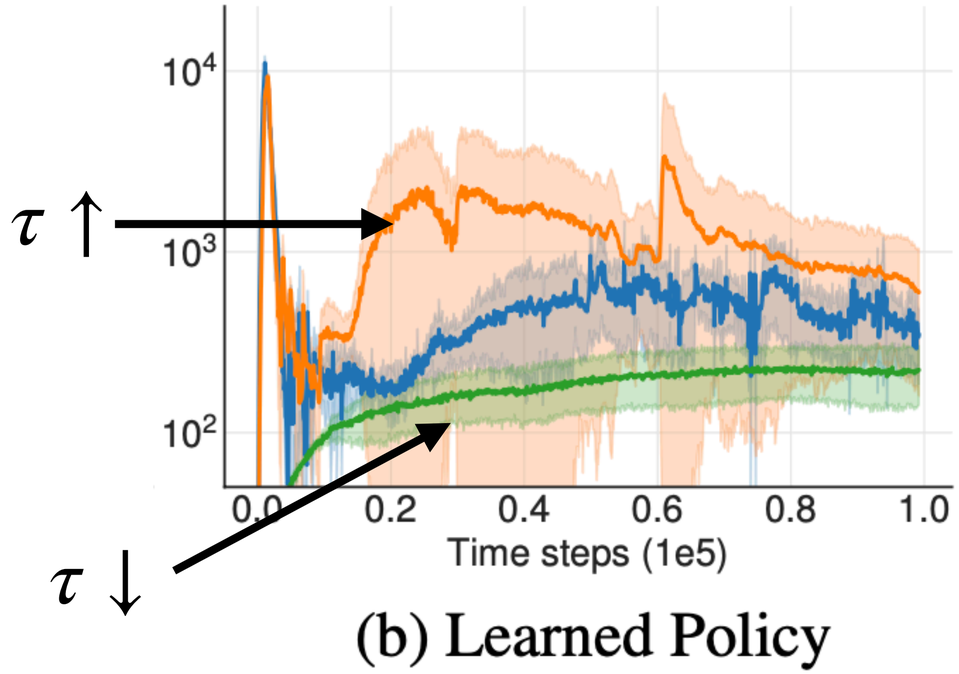


(a) Fixed Policy (b) Learned Policy

TD3

Case study: different update rates for the policy and the target network

- The fluctuation is mainly caused by the fact that both the policy and the Q-function are learning
- As a result, if the target networks changes fast, it may cause the estimated Q-function unstable



(b) Learned Policy

TD3

Delayed updates of the policy and the target network

- The policy network should be updated at a lower frequency
    - This allows the value function to converge and reduce error

- **Delay policy update —** update the policy and the target networks after a fixed number of updates to the critic
    - Less frequent policy updates enable the value estimate to have lower variances
    - This also allows the policy to be updated with a higher quality

TD3

Target policy smooth regularization

- TD3 introduces a technique called "**target policy smooth regularization**" to enhance the robustness of the estimated Q-function
  - **Concept**: Similar actions should have similar Q-values
  - **Method**: Fit the values from a small possible ranges around the target action

$$y = r + \gamma Q(s_{t+1}, \pi_\phi(s_{t+1}) + \epsilon)$$

$$\epsilon \sim clip(\mathcal{N}(0,\sigma), -c, c)$$

TD3

The Pseudo Code of TD3

Algorithm 1 TD3

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ

Initialize target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$

Initialize replay buffer $\mathcal{B}$

for $t = 1$ **to** T **do**

Select action with exploration noise $a \sim \pi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'

Store transition tuple (s, a, r, s') in $\mathcal{B}$

Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$

$\tilde{a} \leftarrow \pi_{\phi'}(s) + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$

$y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$

Update critics $\theta_i \leftarrow \min_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$

if t mod d **then**

Update ϕ by the deterministic policy gradient:

$\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$

Update target networks:

$\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$

$\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

Clipped double Q-learning for actor-critic

Target policy smooth regularization

Delayed Updates of the policy and the target network

TD3

Comparison of TD3 and the other RL methods

- TD3 outperforms some other RL methods in a wide range of MuJoCo environments

Environment	TD3	DDPG	Our DDPG	PPO	TRPO	ACKTR	SAC
HalfCheetah	**9636.95 ± 859.065**	3305.60	8577.29	1795.43	-15.57	1450.46	2347.19
Hopper	**3564.07 ± 114.74**	2020.46	1860.02	2164.70	2471.30	2428.39	2996.66
Walker2d	**4682.82 ± 539.64**	1843.85	3098.11	3317.69	2321.47	1216.70	1283.67
Ant	**4372.44 ± 1000.33**	1005.30	888.77	1083.20	-75.85	1821.94	655.35
Reacher	**-3.60 ± 0.56**	-6.51	**-4.01**	-6.18	-111.43	-4.26	-4.44
InvPendulum	**1000.00 ± 0.00**	**1000.00**	**1000.00**	**1000.00**	985.40	**1000.00**	**1000.00**
InvDoublePendulum	**9337.47 ± 14.96**	**9355.52**	8369.95	8977.94	205.85	9081.92	8487.15

Hindsight Experience Replay (HER)

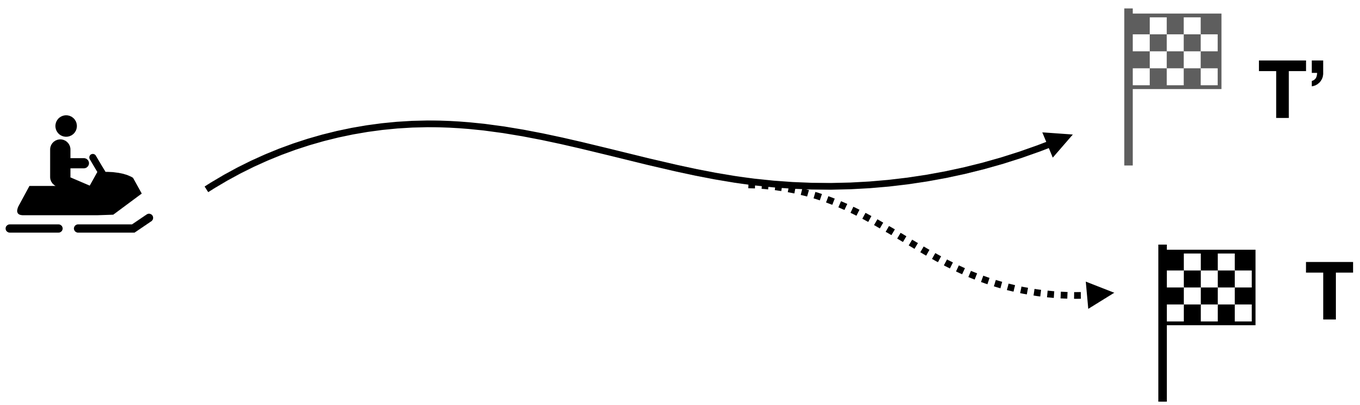
A technique for dealing with sparse reward environments

- HER is designed for handling sparse rewards environments
- HER sets **pseudo goals** for the agent, avoid the complicated reward engineering
- HER performs well on several robotic arm tasks

Hindsight Experience Replay (HER)

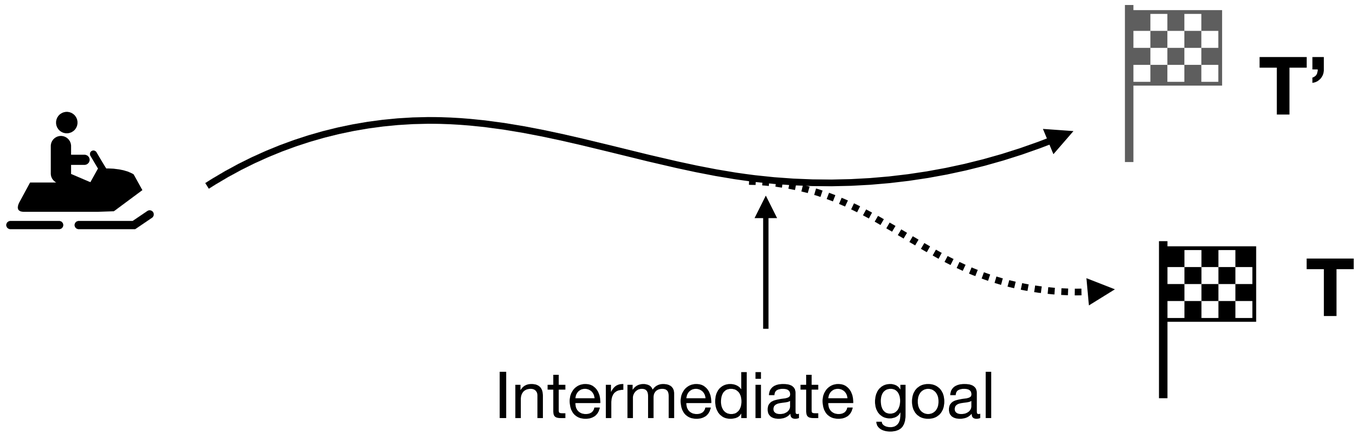
Motivation

- One ability humans is to learn almost as much from achieving an undesired outcome as from the desired one
- Sometimes, although the experience does not reach the goal of task **T**, it still somehow reaches the goal of another task **T'**



Hindsight Experience Replay (HER)

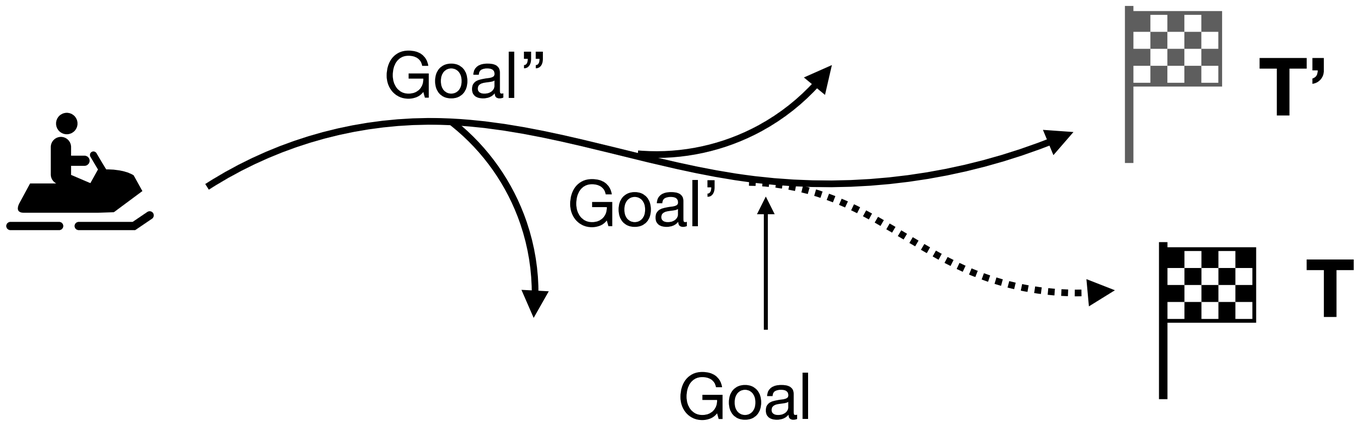
Motivation



- Next time, if you want to reach T or T’, you know you need to reach the ‘**intermediate goal**’ first

Hindsight Experience Replay (HER)

Motivation



- After several trials, you will know a lot of intermediate goals
- Then, you can follow the intermediate goals step by step, and finally reach the true goal

Hindsight Experience Replay (HER)

Ideas in RL

- For RL, it is desired that a learning agent is able to reach some particular results (such that it can collect experiences)
- However, it would fail very often at the beginning of the training phase
- These failed experiences can become useful knowledge for the agent
- This kind of information is called the “intermediate goal”, or the “pseudo goal”

Hindsight Experience Replay (HER)

Algorithm: Initialization phase

- Initialization
  - Select an off-policy RL algorithm
  - m is a mapping function for sampling a goal g from a trajectory
  - $f_g(s)$ is a function for determining if the current state s is the goal state g
  - Initialize the replay buffer and the neural network

Given:

- an off-policy RL algorithm $\mathbb{A}$, ▷ e.g. DQN, DDPG, NAF, SDQN
- a strategy $\mathbb{S}$ for sampling goals for replay, ▷ e.g. $\mathbb{S}(s_0, \ldots, s_T) = m(s_T)$
- a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow \mathbb{R}$. ▷ e.g. $r(s, a, g) = -[f_g(s) = 0]$

Initialize $\mathbb{A}$ ▷ e.g. initialize neural networks

Initialize replay buffer R

Hindsight Experience Replay (HER)

Algorithm

- Use the current policy to generate an whole trajectory

for episode $= 1, M$ **do**
  Sample a goal g and an initial state s_0.
  for $t = 0, T-1$ **do**
    Sample an action a_t using the behavioral policy from $\mathbb{A}$:
      $a_t \leftarrow \pi_b(s_t||g)$ ▷ $||$ denotes concatenation
    Execute the action a_t and observe a new state s_{t+1}
  end for

Hindsight Experience Replay (HER)

Algorithm

---

for $t = 0, T-1$ **do**
  $r_t := r(s_t, a_t, g)$
  Store the transition $(s_t||g,\ a_t,\ r_t,\ s_{t+1}||g)$ in R ▷ standard experience replay
  Sample a set of additional goals for replay $G := \mathbb{S}(\textbf{current episode})$
  for $g' \in G$ **do**
    $r' := r(s_t, a_t, g')$
    Store the transition $(s_t||g',\ a_t,\ r',\ s_{t+1}||g')$ in R ▷ HER
  end for
end for

Hindsight Experience Replay (HER)

Algorithm

- Finally, original update after generating HER data

for $t = 1, N$ **do**
    Sample a minibatch B from the replay buffer R
    Perform one step of optimization using $\mathbb{A}$ and minibatch B
end for

Hindsight Experience Replay (HER)

Experiment

- Three different environments, all have sparse rewards
- The agent only obtains a reward signal when finishing the task

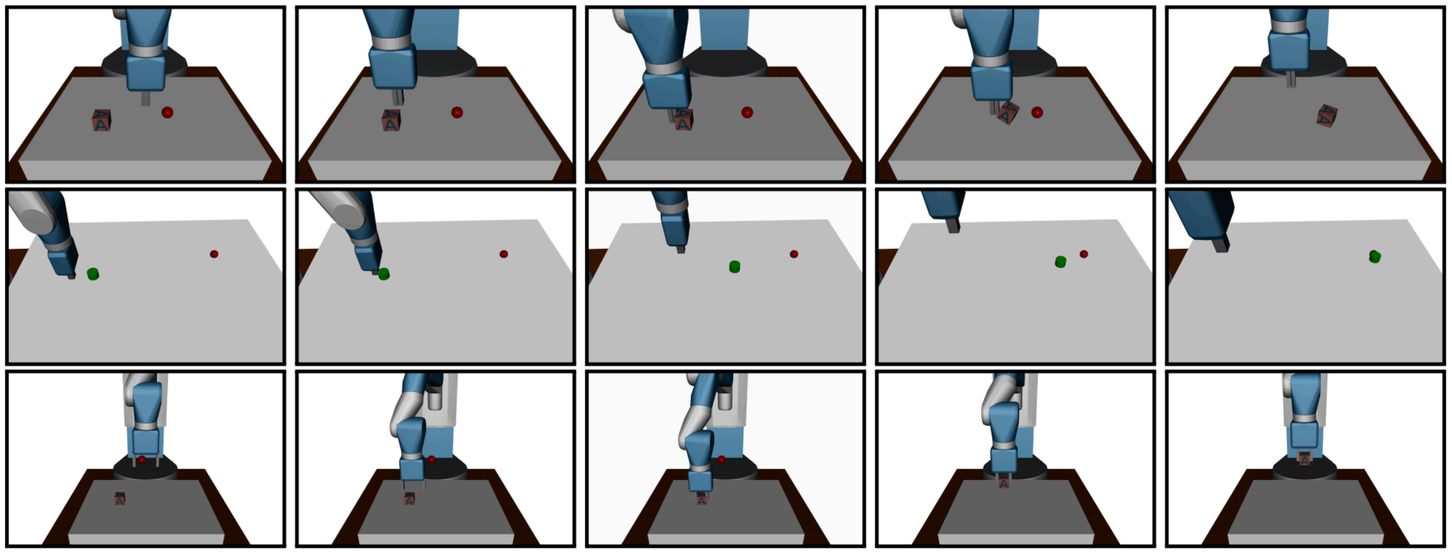
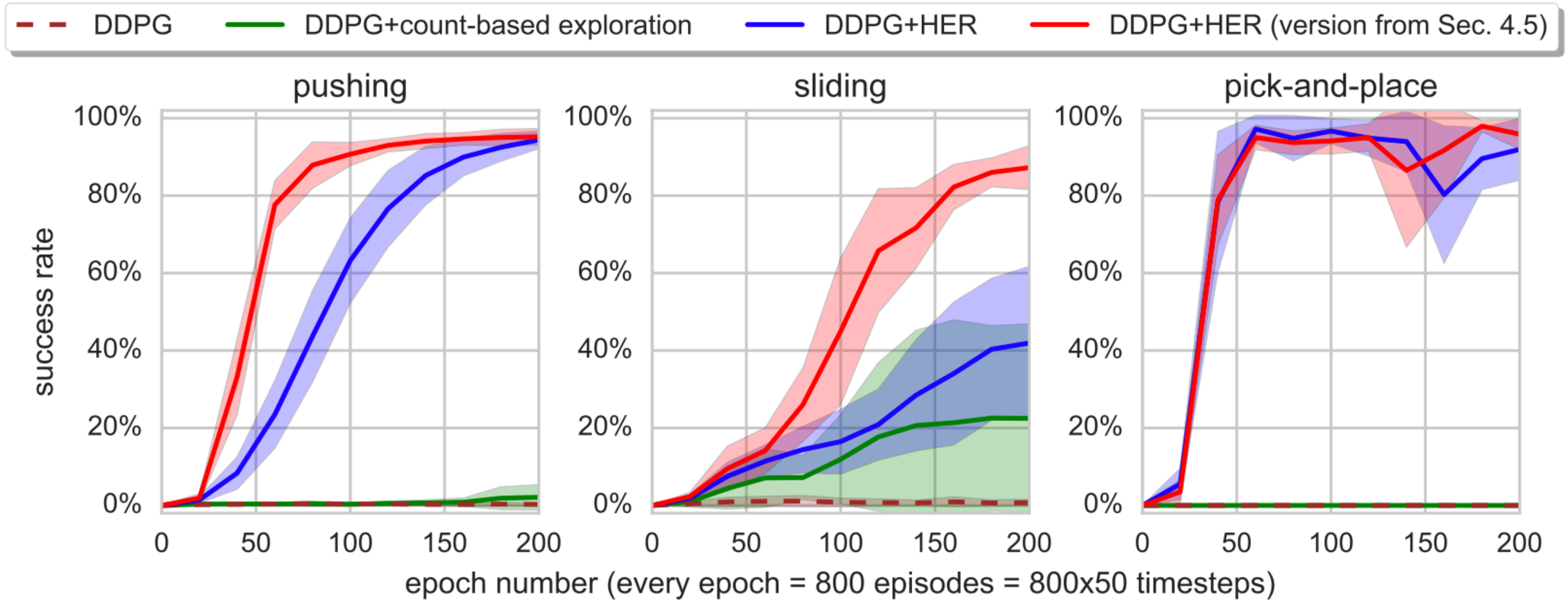


Figure 2: Different tasks: *pushing* (top row), *sliding* (middle row) and *pick-and-place* (bottom row). The red ball denotes the goal position.

Hindsight Experience Replay (HER)

Experiment

- The results show that HER help the training process



Hindsight Experience Replay (HER)

Experiment

- Demo

Figure 7: The pick-and-place policy deployed on the physical robot.

SAC

Soft Actor Critic

- An actor-critic based method
- SAC is based on a stochastic policy method
- SAC is based on the concept of **maximum entropy reinforcement learning**

SAC

Maximum Entropy Reinforcement Learning

- Standard reinforcement learning

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{(s_t,a_t)\sim\pi}[\sum_t R(s_t, a_t)]$$

- Maximum entropy reinforcement learning
  - An entropy term $\alpha\mathcal{H}(\pi(\cdot \mid s_t))$ is introduced to encourage exploration
    - α is a hyper-parameter for controlling how important the entropy term is
    - $\mathcal{H}(\pi(\cdot \mid s_t))$ is the entropy function
    - The policy is trained to maximize **(1) the expected return and (2) entropy of the actions**

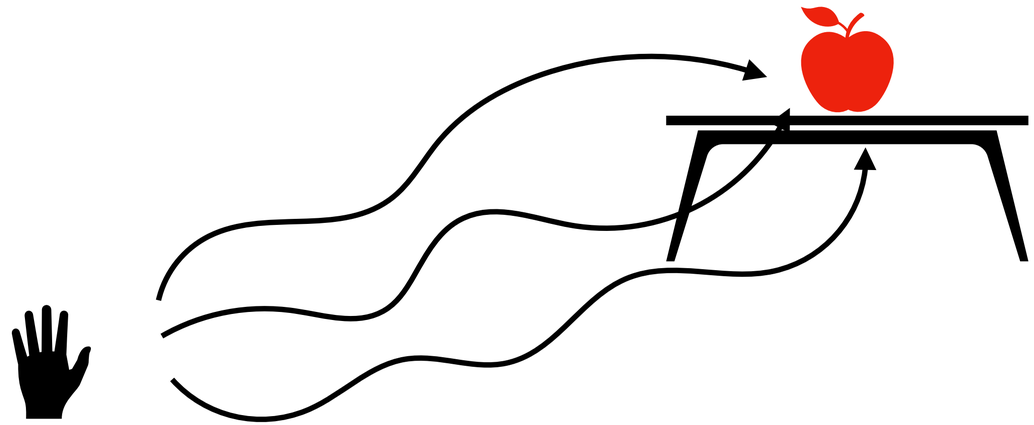
$$\pi^* = \arg\max_{\pi} \mathbb{E}_{(s_t,a_t)\sim\pi}[\sum_t R(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot \mid s_t))]$$

(1) **(2)**

SAC

Maximum entropy reinforcement learning

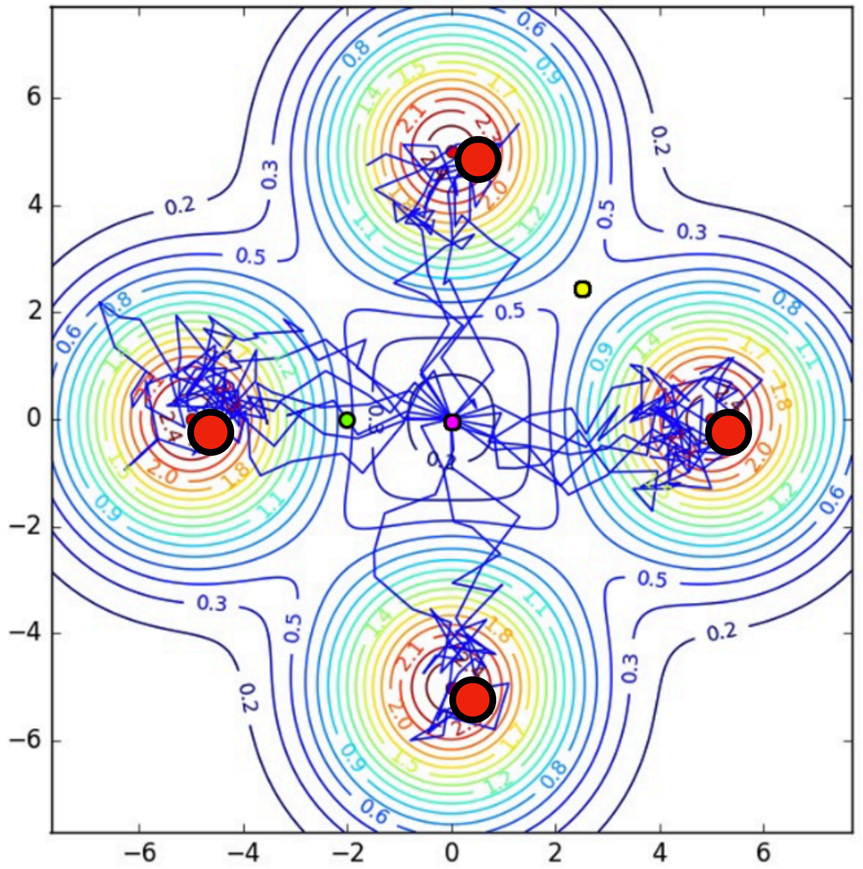
- Entropy term in the objective function for making the policy more exploratory
- SAC enables learning multiple near-optimal strategies
- The learned policy is more robust
  - The policy will not be constrained to certain trajectories
  - The policy will be able to handle unfamiliar states better



SAC

A multi-goal environment example

- The agent is initialized at the center
- The goals are depicted as red dots, and the level curves show the reward.
- Reaching the four goals will get the same reward.

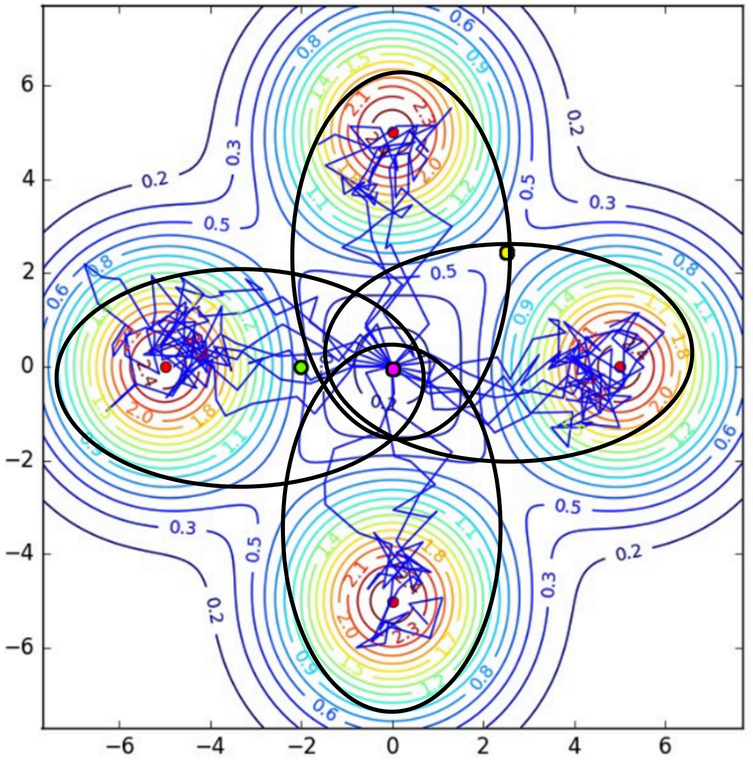


SAC

A multi-goal environment example

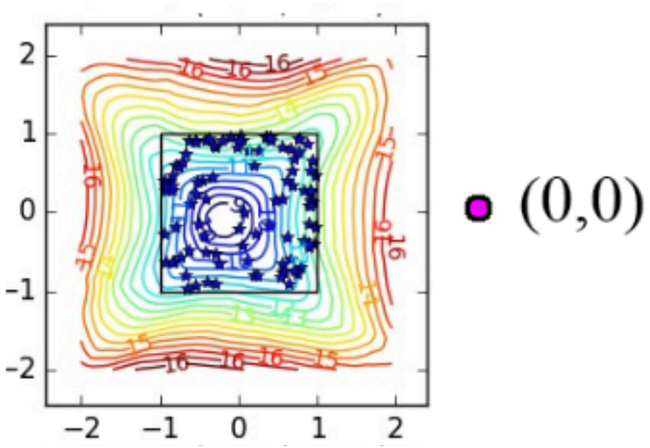
- For standard RL, the policy will only learn a fixed way and always reach the same goal.

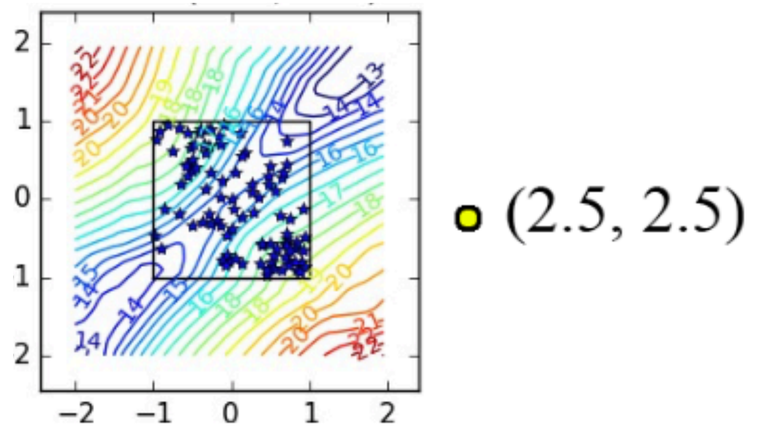
- However, for **Maximum Entropy Reinforcement Learning**, the policy will learn the four different paths and randomly walk



SAC

A multi-goal environment example



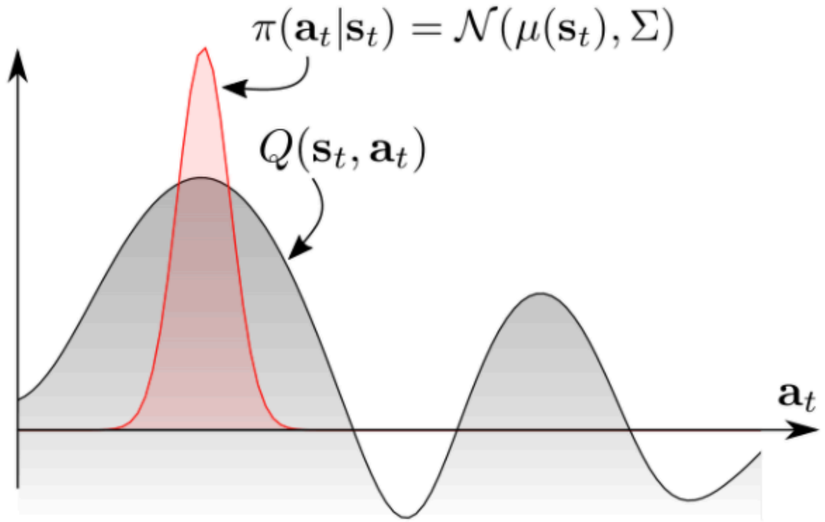


- At the center, the policy chooses multimodal actions, enabling the agent to move to the four different directions
- At the point between two goals, the policy chooses multimodal actions, enabling the agent to move to two different goals randomly.

SAC

Maximum Entropy Reinforcement Learning

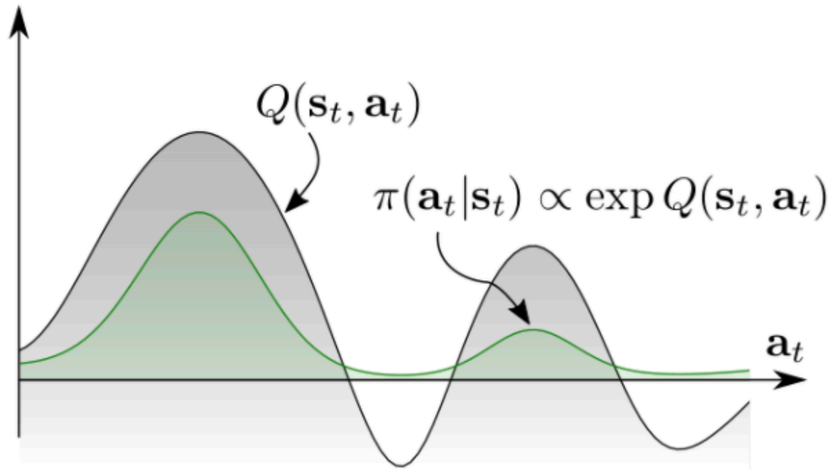
- The Q-value distribution is multimodal
- Standard RL : $\pi(a_t \mid s_t) \propto \mathcal{N}(\mu(s_t), \sigma)$
- The policy will converge to a single peak



SAC

Maximum Entropy Reinforcement Learning

- An energy-based policy assumes that the policy $\pi(a_t \mid s_t) \propto \exp(Q(s_t, a_t))$
- The policy can easily learn multimodal behavior



SAC

The definition of the soft value function

- As a result, we define the soft Q-function as:

$$Q^{\pi}(s_t, a_t) = r_t + \mathbb{E}_{(s_{t+1},..)\sim\pi}[\sum_{i=1}^{\infty} \gamma^i (r_{t+i} + \alpha \mathcal{H}(\pi(\cdot \mid s_{t+i})))]$$

- And the soft value function as:

$$V^{\pi}(s_t) = \mathbb{E}_{(s_t,..)\sim\pi}[\sum_{i=0}^{\infty} \gamma^i (r_{t+i} + \alpha \mathcal{H}(\pi(\cdot \mid s_{t+i})))]$$

Further Reading — Reinforcement Learning with Deep Energy-Based Policies, ArXiv 1702.08165.

SAC

The definition of the soft value function

- The relation between V and Q

$$V^{\pi}(s_t) = \mathbb{E}_{(s_t,..)\sim\pi}[Q^{\pi}(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot \mid s_t))]$$

$$= \mathbb{E}_{(s_t,..)\sim\pi}[Q^{\pi}(s_t, a_t) - \alpha \log \pi(\cdot \mid s_t))]$$

SAC

Update Soft Value function

- The Bellman equation :

$$Q^{\pi}(s_t, a_t) = r_t + \mathbb{E}_{(s_{t+1},..)\sim\pi}[\sum_{i=1}^{\infty} \gamma^i (r_{t+i} + \alpha \mathcal{H}(\pi(\cdot \mid s_{t+i})))]$$

Unrolled

$$\gamma(r_{t+1} + \underbrace{\sum_{i=2}^{\infty} \gamma^i (r_{t+i} + \alpha \mathcal{H}(\pi(\cdot \mid s_{t+i})))}_{Q^{\pi}(s_{t+1}, a_{t+1})}) + \mathcal{H}(\pi(\cdot \mid s_{t+1}))$$

$$= r_t + \mathbb{E}_{(s_{t+1},..)\sim\pi}[\gamma(Q^{\pi}(s_{t+1}, a_{t+1}) + \alpha \mathcal{H}(\pi(\cdot \mid s_{t+1})))]$$

SAC

Update Value function

Lemma 1 (Soft Policy Evaluation). *Consider the soft Bellman backup operator $\mathcal{T}^\pi$ in Equation 2 and a mapping $Q^0 : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define $Q^{k+1} = \mathcal{T}^\pi Q^k$. Then the sequence Q^k will converge to the soft Q-value of π as $k \rightarrow \infty$.*

- The Bellman equation :

$$Q^\pi(s_t, a_t) = r_t + \mathbb{E}_{(s_{t+1},..)\sim\pi}[\gamma(Q^\pi(s_{t+1}, a_{t+1}) + \alpha\mathcal{H}(\pi(\cdot \mid s_{t+1})))]$$

$$V^\pi(s_{t+1})$$

By page. 50

$$Q^\pi(s_t, a_t) = r_t + \mathbb{E}_{(s_{t+1},..)\sim\pi}[\gamma V(s_{t+1})]$$

SAC

Update policy

Lemma 2 (Soft Policy Improvement). *Let* $\pi_{\text{old}} \in \Pi$ *and let* π_{new} *be the optimizer of the minimization problem defined in Equation 4. Then* $Q^{\pi_{\text{new}}}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^{\pi_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t)$ *for all* $(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{S} \times \mathcal{A}$ *with* $|\mathcal{A}| < \infty$.

- Note that for multimodal behavior, we need to $\pi \propto \exp(Q)$
- Therefore, we expect the new π to be close to the the distribution of $\exp(Q)$

$$\pi_{new} = \arg\min_{\pi' \in \Pi} D_{KL}\left(\pi'(\,.\,|\,s_t) \,\middle\|\, \frac{\exp(Q^{\pi_{old}}(s_t, .\,))}{Z^{\pi_{old}}(s_t)} \right)$$

Z, the normalization function for the distribution

SAC

Algorithm

- Three things need to be maintain and update
  - (1) State Value function $V^{\pi}(s_t)$ parameterize with ψ
  - (2) Q-Value function $Q^{\pi}(s_t, a_t)$ parameterize with θ
  - (3) Policy $\pi(\,.\,|\,s_t)$ parameterize with ϕ
- Use data samples from replay buffer $\mathbf{D}$

SAC

Value function

$$V^{\pi}(s_t) = \mathbb{E}_{(s_t,..)\sim\pi}[Q^{\pi}(s_t, a_t) - \alpha \log(\pi(\cdot \mid s_t))]$$

- (1) State Value function $V^{\pi}(s_t)$, with MSE

$$J_V(\psi) = \mathbb{E}_{s_t\sim D}\left[\frac{1}{2}(V_\psi(s_t) - \mathbb{E}_{a_t\sim\pi_\phi}[Q_\theta(s_t, a_t) - \log \pi_\phi(a_t \mid s_t)])^2\right]$$

By page. 51

$$\nabla_\psi J_V(\psi) = \nabla_\psi V_\psi(s_t)(V_\psi(s_t) - Q_\theta(st, a_t) + \log \pi_\phi(a_t \mid s_t))$$

SAC

Q-Value function

$$Q^{\pi}(s_t, a_t) = r_t + \mathbb{E}_{(s_{t+1},..)\sim\pi}[\gamma V(s_{t+1})]$$

- (2) Q-Value function $Q^{\pi}(s_t, a_t)$, with MSE

$$J_Q(\theta) = \mathbb{E}_{s_t,a_t\sim D}\left[\frac{1}{2}(Q_\theta(s_t, a_t) - (r_t + \gamma\mathbb{E}_{s_{t+1}\sim p}[V_{\bar{\psi}}(s_{t+1})])^2\right]$$

By page. 52

$$\nabla_\theta J_Q(\theta) = \nabla_\theta Q_\theta(s_t, a_t)\Big(Q_\theta(s_t, a_t) - r_t - \gamma V_{\bar{\psi}}(s_{t+1})\Big)$$

$V_{\bar{\psi}}$ Is the target function of V_ψ

SAC

Policy

- (3) Policy $\pi(\,.\,|\,s_t)$, with

$$D_{\mathrm{KL}}(P\|Q) = -\sum_i P(i)\ln\frac{Q(i)}{P(i)}.$$

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim D}\left[D_{KL}\left(\pi_\phi(\,.\,|\,s_t)\,||\,\frac{\exp(Q_\theta(s_t,\,.\,))}{Z_\theta(s_t)}\right)\right]$$

$$= \mathbb{E}_{s_t\sim D}\left[\mathbb{E}_{a_t\sim\pi_\phi}\left[\log\left(\frac{\pi_\phi(a_t\,|\,s_t)}{\exp(Q_\theta(s_t,a_t)-\log Z_\theta(s_t))}\right)\right]\right]$$

$$= \mathbb{E}_{s_t\sim D}\left[\mathbb{E}_{a_t\sim\pi_\phi}[\log\pi_\phi(a_t\,|\,s_t) - Q_\theta(s_t,a_t) + \log Z_\theta(s_t)]\right]$$

SAC

Policy

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim D, a_t \sim \pi_\phi}\left[\log \pi_\phi(a_t \mid s_t) - Q_\theta(s_t, a_t) + \log Z_\theta(s_t)\right]$$

· By the reparameterization trick, $a_t = f_\phi(\epsilon_t; s_t)$, ϵ_t is an input noise sampled from some fixed distribution (e.g. Gaussian)

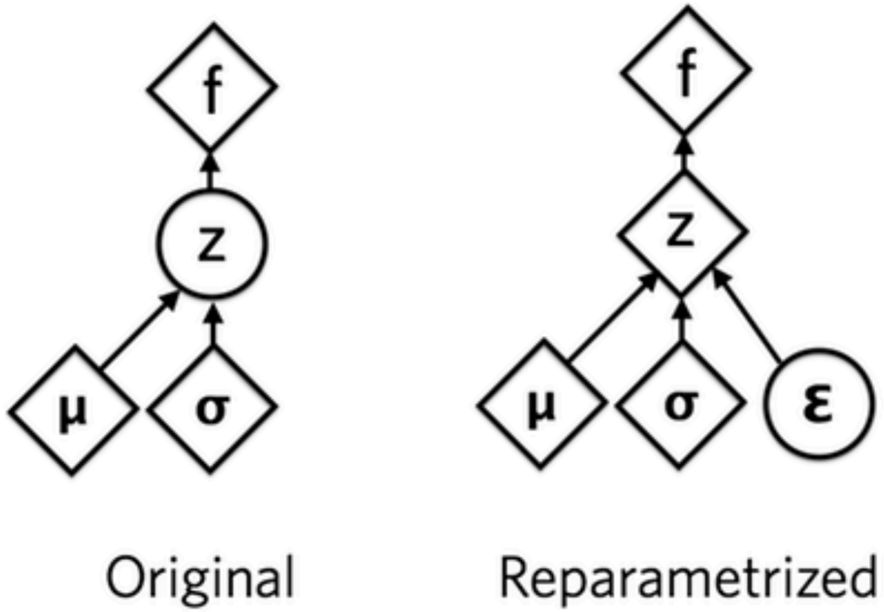
· Finally rewrite J as

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim D, \epsilon_t \sim N}\left[\log \pi_\phi(f_\phi(\epsilon_t; s_t) \mid s_t) - Q_\theta(s_t, f_\phi(\epsilon_t; s_t)) - \log Z_\theta)\right]$$

SAC

Reparameterization trick

- The case of a VAE



SAC

Policy

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim D, \epsilon_t \sim N}\Big[\log \pi_\phi(f_\phi(\epsilon_t; s_t) \mid s_t) - Q_\theta(s_t, f_\phi(\epsilon_t; s_t)) - \log Z_\theta)\Big]$$

- The gradient is :

$$\begin{aligned}\nabla_\theta \mathbb{E}_{q_\theta(z)}[f_\theta(z)] &= \nabla_\theta \int q_\theta(z) \cdot f_\theta(z)\, dz \\ &= \int q_\theta(z) \cdot \nabla_\theta\, f_\theta(z)\, dz + \int \nabla_\theta\, q_\theta(z) \cdot\, f_\theta(z)\, dz \\ &= \mathbb{E}_{q_\theta(z)}\Big[\frac{\partial f_\theta(z)}{\partial \theta}\Big] + \mathbb{E}_{q_\theta(z)}\Big[\frac{df_\theta(z)}{dz} \cdot \frac{dz}{d\theta}\Big]\end{aligned}$$

$$\nabla_\phi J_\pi(\phi) = \nabla_\phi \log \pi_\phi(a_t \mid s_t) + \Big(\nabla_{a_t} \log \pi_\phi(a_t \mid s_t) - \nabla_{a_t} Q_\theta(s_t, a_t)\Big) \nabla_\phi f_\phi(\epsilon_t; s_t)$$

SAC

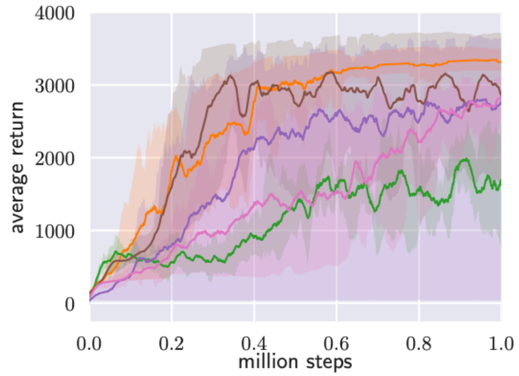
The Pseudo Code of SAC

Algorithm 1 Soft Actor-Critic

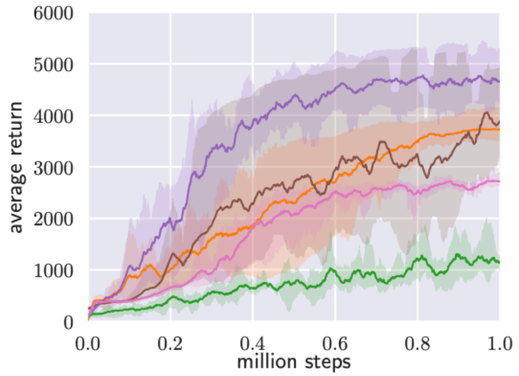
Initialize parameter vectors ψ, $\bar{\psi}$, θ, ϕ.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$
  end for
end for

SAC

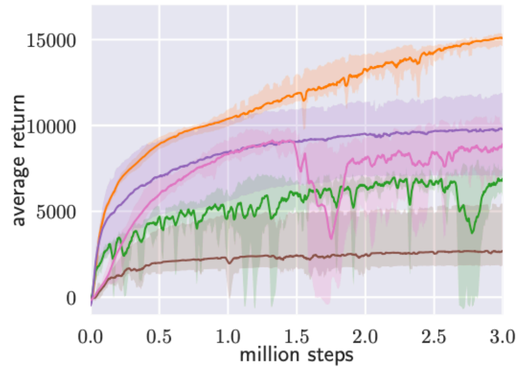
Comparison of SAC and the other RL methods



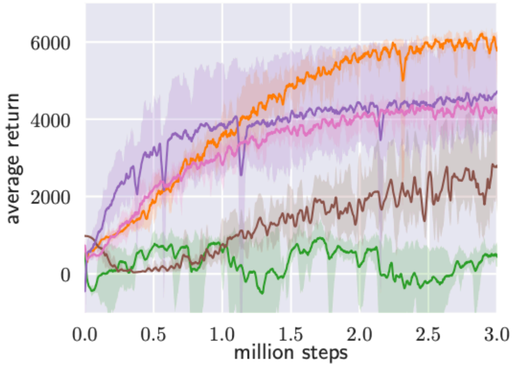
(a) Hopper-v1



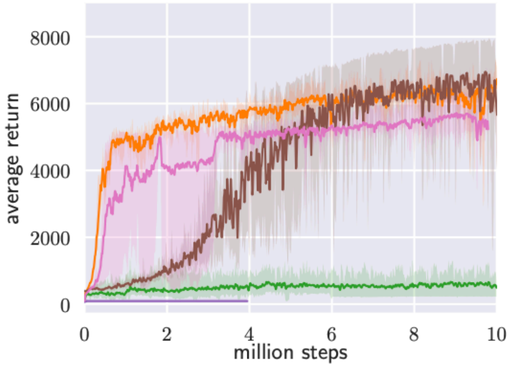
(b) Walker2d-v1



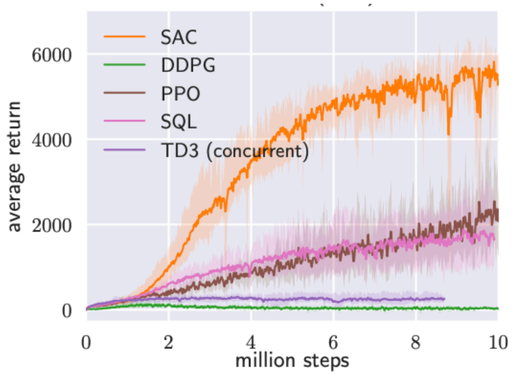
(c) HalfCheetah-v1



(d) Ant-v1



(e) Humanoid-v1



(f) Humanoid (rllab)

ありがとう
ございます