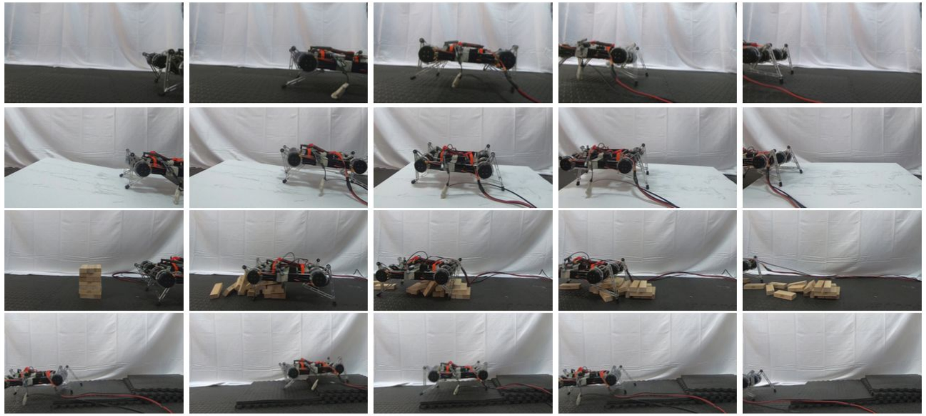
Soft Actor-Critic



Zikun Chen, Minghan Li
Jan. 28, 2020

Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, Sergey Levine

Outline

- Problem: Sample Efficiency
- Solution: Off-Policy Learning
    - On-Policy vs Off-Policy
    - RL Basics Recap
    - Off-Policy Learning Algorithms
- Problem: Robustness
- Solution: Maximum Entropy RL
    - Definition (Control as Inference)
    - Soft Policy Iteration
    - Soft Actor-Critic

Contributions

- An off-policy maximum entropy deep reinforcement learning algorithm
  - Sample-efficient
  - Robustness to noise, random seed and hyperparameters
  - Scale to high-dimensional observation/action space
- Theoretical Results
  - Theoretical framework of soft policy iteration
  - Derivation of soft-actor critic algorithm
- Empirical Results
  - SAC outperforms SOTA model-free deep RL methods, including DDPG, PPO and Soft Q-learning, in terms of the policy's optimality, sample complexity and stability.

Outline

- Problem: **Sample Efficiency**
- Solution: Off-Policy Learning
  - On-Policy vs Off-Policy
  - RL Basics Recap
  - Off-Policy Learning Algorithms

Main Problem: Sample Inefficiency

- Number of times the agent must interact with the environment in order to learn a task

- Good sample complexity is the first prerequisite for successful skill acquisition.

- Learning skills in the real world can take a substantial amount of time
  - can get damaged through trial and error

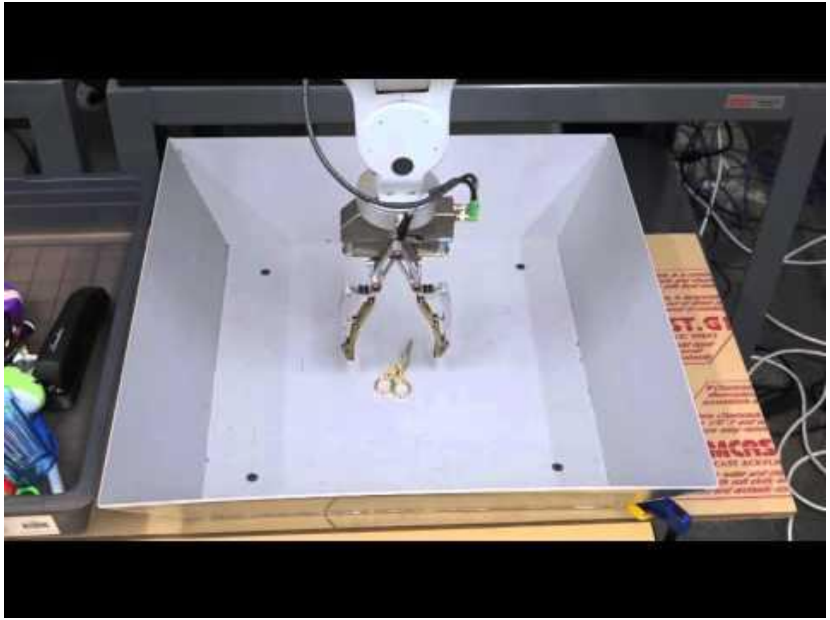
Main Problem: Sample Inefficiency

- "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection", Levine et al., 2016
    - 14 robot arms learning to grasp in parallel
    - objects started being picked up at around 20,000 grasps

https://spectrum.ieee.org/automaton/robotics/artificial-intelligence/google-large-scale-robotic-grasping-project

Main Problem: Sample Inefficiency

> *Observing the behavior of the robot after over 800,000 grasp attempts, which is equivalent to about 3000 robot-hours of practice, we can see the beginnings of intelligent reactive behaviors*



https://www.youtube.com/watch?v=cXaic_k80uM

Main Problem: Sample Inefficiency

- Solution?

- Off-Policy Learning!

Background: On-Policy vs. Off-Policy

- On-policy learning: use the deterministic outcomes or samples from the target policy to train the algorithm
    - has low sample efficiency (TRPO, PPO, A3C)
    - require new samples to be collected for nearly every update to the policy
    - becomes extremely expensive when the task is complex

- Off-policy methods: training on a distribution of transitions or episodes produced by a different behavior policy rather than that produced by the target policy
    - Does not require full trajectories and can reuse any past episodes (experience replay) for much better sample efficiency
    - relatively straightforward for Q-learning based methods

Background: Bellman Equation

- Value Function: How good is a state?

$$
\begin{aligned}
V(s) &= \mathbb{E}[G_t | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots) | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\
&= \mathbb{E}[\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{temporal difference target}} | S_t = s]
\end{aligned}
$$

- Similarly, for Q-Function: How good is a state-action pair?

$$
\begin{aligned}
Q(s, a) &= \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \mathbb{E}[R_{t+1} + \gamma \mathbb{E}_{a \sim \pi} Q(S_{t+1}, a) \mid S_t = s, A_t = a]
\end{aligned}
$$

Background: Value-Based Method

- $\ldots, S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, \ldots$ (on-policy):

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

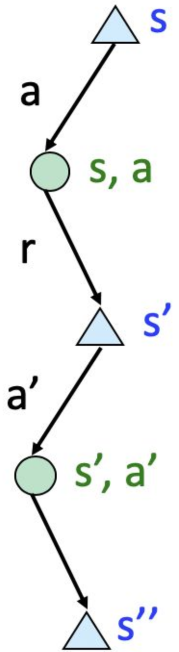
- Q-Learning (off-policy)

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a \in \mathcal{A}} Q(S_{t+1}, a) - Q(S_t, A_t))$$

- DQN, Minh et al., 2015

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s') \sim U(D)} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right)^2 \right]$$

  - Function Approximation
  - Experience Replay: samples randomly drawn from replay memory
  - Doesn't scale to continuous action space



Background: Policy-Based Method (Actor-Critic)

- **Critic**: updates value function parameters w and depending on the algorithm it could be action-value $Q(a|s; w)$ or state-value $V(s; w)$.
- **Actor**: updates policy parameters θ, in the direction suggested by the critic, $\pi(a|s; \theta)$.

$$\nabla \mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}[\nabla \ln \pi(a|s, \theta) Q_\pi(s, a)]$$

policy gradient

$$\theta \leftarrow \theta + \alpha_\theta Q(s, a; w) \nabla_\theta \ln \pi(a|s; \theta)$$

update actor

$$G_{t:t+1} = r_t + \gamma Q(s', a'; w) - Q(s, a; w)$$

correction for action-value

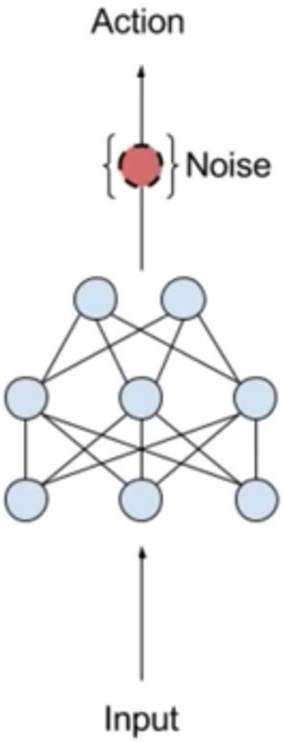
$$w \leftarrow w + \alpha_w G_{t:t+1} \nabla_w Q(s, a; w).$$

update critic

https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html#actor-critic

Prior Work: DDPG

- DDPG = DQN + DPG (Lillicrap et al., 2015)
    - off-policy actor-critic method that learns a deterministic policy in continuous domain
    - exploration noise added to the deterministic policy when select action
    - difficult to stabilize and brittle to hyperparameters (Duan et al., 2016, Henderson et al., 2017)
    - unscalable to complex tasks with high dimensions (Gu et al., 2017)



https://www.youtube.com/watch?v=zR11FLZ-O9M&t=2145s

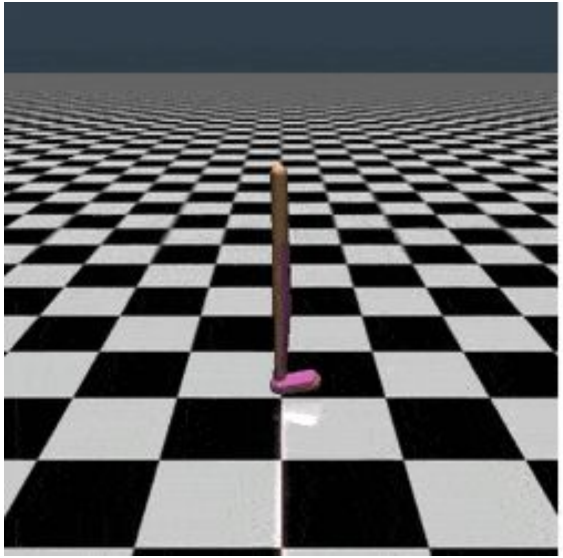
Outline

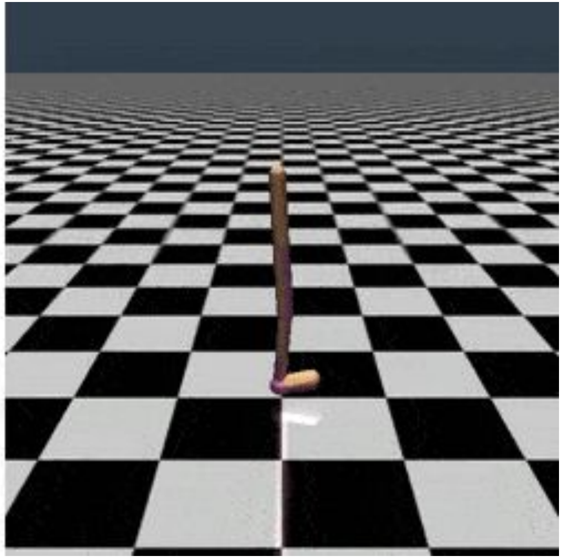
- Problem: Sample Efficiency
- Solution: Off-Policy Learning
  - On-Policy vs Off-Policy
  - RL Basics Recap
  - Off-Policy Learning Algorithms

- **Problem: Robustness**
- Solution: Maximum Entropy RL
  - Definition (Control as Inference)
  - Soft Policy Iteration
  - Soft Actor-Critic

Main Problems: Robustness

- Training is sensitive to randomness in the environment, initialization of the policy and the algorithm implementation

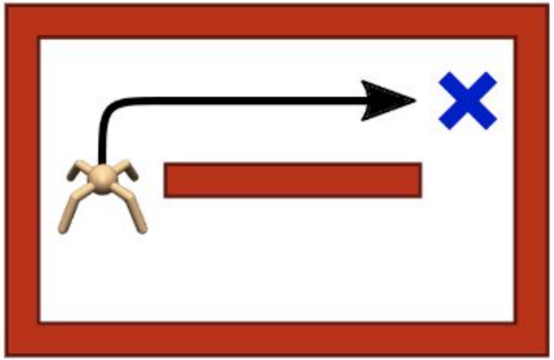
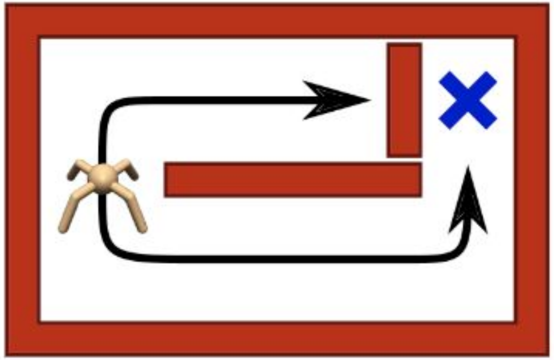




https://gym.openai.com/envs/Walker2d-v2/

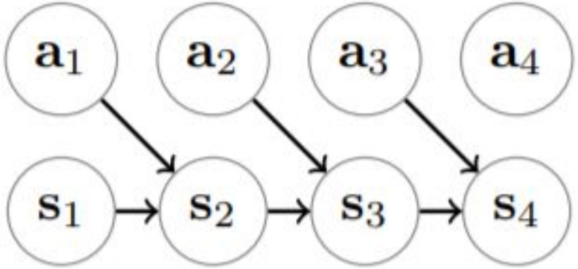
Main Problems: Robustness

- Knowing only one way to act makes agents vulnerable to environmental changes that are common in the real-world

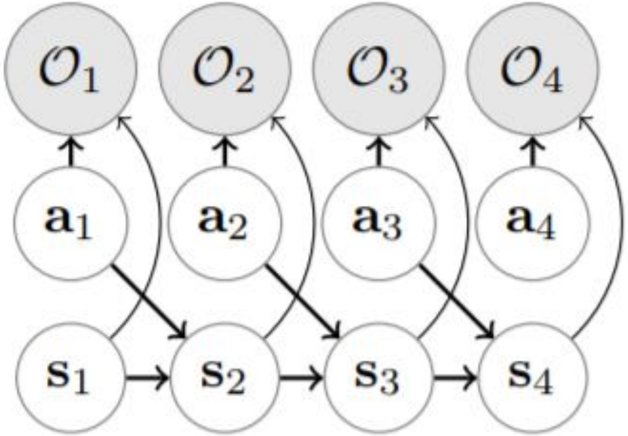



https://bair.berkeley.edu/blog/2017/10/06/soft-q-learning/

Background: Control as Inference



Traditional Graph of MDP



Graphical Model with Optimality Variables

Background: Control as Inference

Normal trajectory distribution

$$p(\tau) = p(\mathbf{s}_1, \mathbf{a}_t, \ldots, \mathbf{s}_T, \mathbf{a}_T | \theta) = p(\mathbf{s}_1) \prod_{t=1}^{T} p(\mathbf{a}_t | \mathbf{s}_t, \theta) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t).$$

Posterior trajectory distribution

$$p(\mathcal{O}_t = 1 | \mathbf{s}_t, \mathbf{a}_t) = \exp(r(\mathbf{s}_t, \mathbf{a}_t)).$$

$$\begin{aligned} p(\tau | \mathbf{o}_{1:T}) \propto p(\tau, \mathbf{o}_{1:T}) &= p(\mathbf{s}_1) \prod_{t=1}^{T} p(\mathcal{O}_t = 1 | \mathbf{s}_t, \mathbf{a}_t) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t) \\ &= p(\mathbf{s}_1) \prod_{t=1}^{T} \exp(r(\mathbf{s}_t, \mathbf{a}_t)) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t) \\ &= \left[p(\mathbf{s}_1) \prod_{t=1}^{T} p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t) \right] \exp\left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right). \end{aligned}$$

Background: Control as Inference

Variational Inference

$$D_{\mathrm{KL}}(\hat{p}(\tau)\|p(\tau)) = -E_{\tau\sim\hat{p}(\tau)}[\log p(\tau) - \log \hat{p}(\tau)].$$

$$\begin{aligned}
-D_{\mathrm{KL}}(\hat{p}(\tau)\|p(\tau)) = E_{\tau\sim\hat{p}(\tau)}\Bigg[&\log p(\mathbf{s}_1) + \sum_{t=1}^{T}\left(\log p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t) + r(\mathbf{s}_t,\mathbf{a}_t)\right) - \\
&\log p(\mathbf{s}_1) - \sum_{t=1}^{T}\left(\log p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t) + \log \pi(\mathbf{a}_t|\mathbf{s}_t)\right)\Bigg] \\
= E_{\tau\sim\hat{p}(\tau)}&\left[\sum_{t=1}^{T} r(\mathbf{s}_t,\mathbf{a}_t) - \log \pi(\mathbf{a}_t|\mathbf{s}_t)\right] \\
= \sum_{t=1}^{T}& E_{(\mathbf{s}_t,\mathbf{a}_t)\sim\hat{p}(\mathbf{s}_t,\mathbf{a}_t))}[r(\mathbf{s}_t,\mathbf{a}_t) - \log \pi(\mathbf{a}_t|\mathbf{s}_t)] \\
= \sum_{t=1}^{T}& E_{(\mathbf{s}_t,\mathbf{a}_t)\sim\hat{p}(\mathbf{s}_t,\mathbf{a}_t))}[r(\mathbf{s}_t,\mathbf{a}_t)] + E_{\mathbf{s}_t\sim\hat{p}(\mathbf{s}_t)}[\mathcal{H}(\pi(\mathbf{a}_t|\mathbf{s}_t))].
\end{aligned}$$

Background: Max Entropy RL

Conventional RL Objective - Expected Reward

$$\sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_t,\mathbf{a}_t)\right]$$

Maximum Entropy RL Objective - Expected Reward + Entropy of Policy

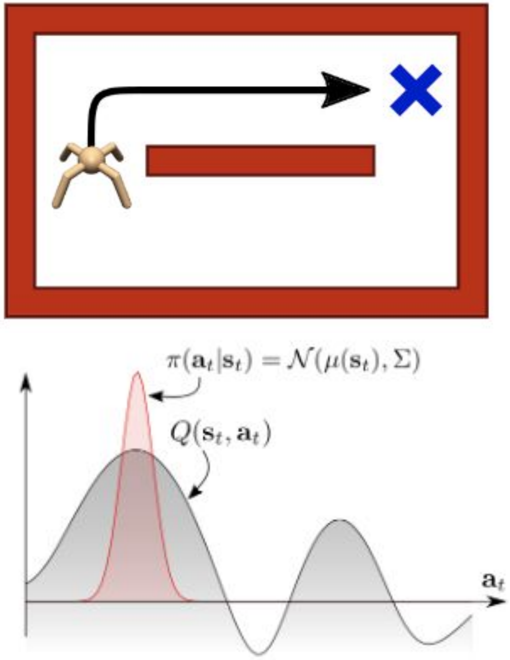
$$\sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_t,\mathbf{a}_t) + \alpha\boxed{\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t))}\right]$$

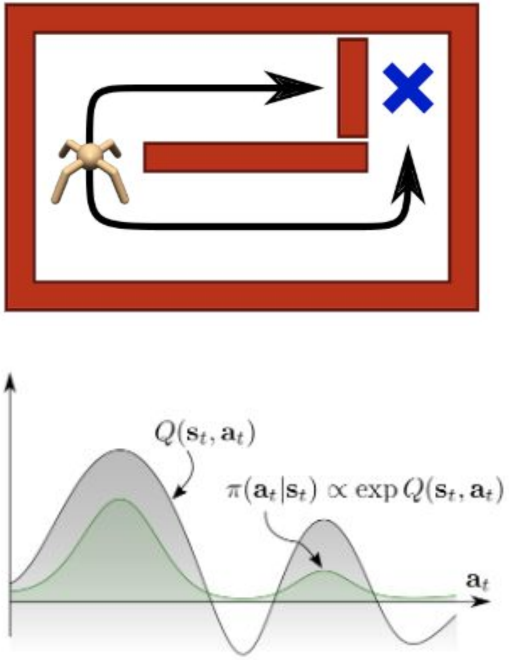
Entropy of a RV x

$$H(P) = \mathop{\mathrm{E}}_{x\sim P}[-\log P(x)]$$

Max Entropy RL

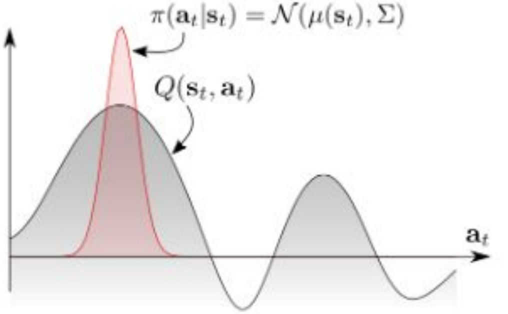
- MaxEnt RL agent can capture different modes of optimality to improve robustness against environmental changes

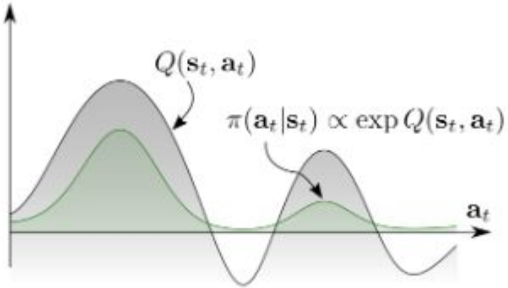




https://bair.berkeley.edu/blog/2017/10/06/soft-q-learning/

Max Entropy RL





$$
\begin{aligned}
&\min_\pi D_{KL}[\pi(\cdot|s_0)||\exp(Q(s_0,\cdot))] \\
&= \min_\pi E_\pi[\log \frac{\pi(a_0|s_0)}{\exp(Q(s_0,a_0))}] \\
&= \max_\pi E_\pi[Q(s_0,a_0) - \log\pi(a_0|s_0)] \\
&= \max_\pi E_\pi[\sum_t r(s_t,a_t) + \mathcal{H}(\pi(\cdot|s_0))|s_0] \\
&= J_{MaxEnt}(\pi(\cdot|s_0))
\end{aligned}
$$

Prior Work: Soft Q-Learning

- Soft Q-Learning (Haarnoja et al., 2017)
  - off-policy algorithms under MaxEnt RL objective
  - Learns Q* directly
  - sample policy from exp(Q*) is intractable for continuous actions
  - use approximate inference methods to sample
    - Stein variational gradient descent
  - not true actor-critic

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a \in \mathcal{A}} Q(S_{t+1}, a) - Q(S_t, A_t))$$

$$Q_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow r_t + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1} \sim p_{\mathbf{s}}} \left[V_{\mathrm{soft}}(\mathbf{s}_{t+1}) \right], \ \forall \mathbf{s}_t, \mathbf{a}_t$$

$$V_{\mathrm{soft}}(\mathbf{s}_t) \leftarrow \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q_{\mathrm{soft}}(\mathbf{s}_t, \mathbf{a}') \right) d\mathbf{a}', \forall \mathbf{s}_t$$

SAC: Contributions

- One of the most efficient model-free algorithms
  - SOTA off-policy
  - well suited for real world robotics learning
- Can learn stochastic policy on continuous action domain
- Robust to noise

- Ingredients:
  - Actor-critic architecture with seperate policy and value function networks
  - Off-policy formulation to reuse of previously collected data for efficiency
  - Entropy-constrained objective to encourage stability and exploration

Soft Policy Iteration: Policy Evaluation

- policy evaluation: compute value of π according to Max Entropy RL Objective

- modified Bellman backup operator T:

$$\mathcal{T}^\pi Q(\mathbf{s}_t, \mathbf{a}_t) \triangleq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \, \mathbb{E}_{\mathbf{s}_{t+1} \sim p} \left[\boxed{V(\mathbf{s}_{t+1})} \right]$$

$$V(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi} \left[Q(\mathbf{s}_t, \mathbf{a}_t) - \boxed{\alpha \log \pi(\mathbf{a}_t | \mathbf{s}_t)} \right]$$

- Lemma 1: Contraction Mapping for Soft Bellman Updates

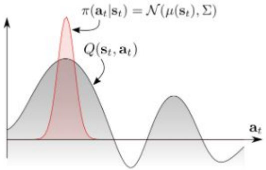
$$Q^{k+1} = \mathcal{T}^\pi Q^k$$ converges to the soft Q-function of π

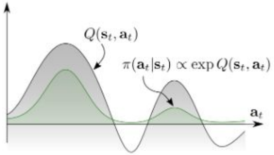
Soft Policy Iteration: Policy Improvement

- policy improvement: update policy towards the exponential of the new soft Q-function

- modified Bellman backup operator T:
  - choose tractable family of distributions big Π
  - choose KL divergence to project the improved policy into big Π

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi'(\,\cdot\,|\mathbf{s}_t) \,\middle\|\, \frac{\exp\left(\frac{1}{\alpha} Q^{\pi_{\text{old}}}(\mathbf{s}_t,\,\cdot\,)\right)}{Z^{\pi_{\text{old}}}(\mathbf{s}_t)}\right)$$





- Lemma 2

$$Q^{\pi_{\text{new}}}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^{\pi_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t)$$ for any state action pair

Soft Policy Iteration

- soft policy iteration: soft policy evaluation <-> soft policy improvement

- Theorem 1: Repeated application of soft policy evaluation and soft policy improvement from any policy $\pi \in \Pi$ converges to the optimal MaxEnt policy among all policies in Π
  - exact form applicable only in discrete case
  - need function approximation to represent Q-values in continuous domains
  - -> Soft Actor-Critic (SAC)!

SAC

$Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$

parameterized soft Q-function

- e.g.neural network

$\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$

parameterized tractable policy

- e.g. Gaussian with mean and covariances given by neural networks

$J_Q(\theta)$ $\hat{\nabla}_\theta J_Q(\theta)$

soft Q-function objective and its stochastic gradient wrt its parameters

$J_\pi(\phi)$ $\hat{\nabla}_\phi J_\pi(\phi)$

policy objective and stochastic gradient wrt its parameters

SAC: Objectives and Optimization

- Critic - Soft Q-function
  - minimize square error
  - $\bar{\theta}$ exponential moving average of soft Q-function weights to stabilize training (DQN)

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

$$V(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi} \left[Q(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi(\mathbf{a}_t | \mathbf{s}_t) \right]$$

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\mathbf{a}_t, \mathbf{s}_t) \left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \left(Q_{\bar{\theta}}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - \alpha \log \left(\pi_\phi(\mathbf{a}_{t+1} | \mathbf{s}_{t+1}) \right) \right) \right) \right)$$

SAC: Objectives and Optimization

- Actor - Policy $\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}} \left(\pi'(\,\cdot\,|\mathbf{s}_t) \,\middle\|\, \frac{\exp\left(\frac{1}{\alpha} Q^{\pi_{\text{old}}}(\mathbf{s}_t, \,\cdot\,)\right)}{Z^{\pi_{\text{old}}}(\mathbf{s}_t)} \right)$

- multiply by alpha and ignoring the normalization Z

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\mathbb{E}_{\mathbf{a}_t \sim \pi_\phi} \left[\alpha \log\left(\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\right) - Q_\theta(\mathbf{s}_t, \mathbf{s}_t) \right] \right]$$

- reparameterize with neural network f $\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t)$
  - epsilon: input noise vector, sampled from a fixed distribution (spherical Gaussian)

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}} \left[\alpha \log \pi_\phi(f_\phi(\epsilon_t; \mathbf{s}_t)|\mathbf{s}_t) - Q_\theta(\mathbf{s}_t, f_\phi(\epsilon_t; \mathbf{s}_t)) \right]$$

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \alpha \log\left(\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\right) + \left(\nabla_{\mathbf{a}_t} \alpha \log\left(\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\right) - \nabla_{\mathbf{a}_t} Q(\mathbf{s}_t, \mathbf{a}_t)\right) \nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t)$$

- Unbiased gradient estimator that extends DDPG stype policy gradients to any tractable stochastic policy

SAC: Algorithm

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

Note

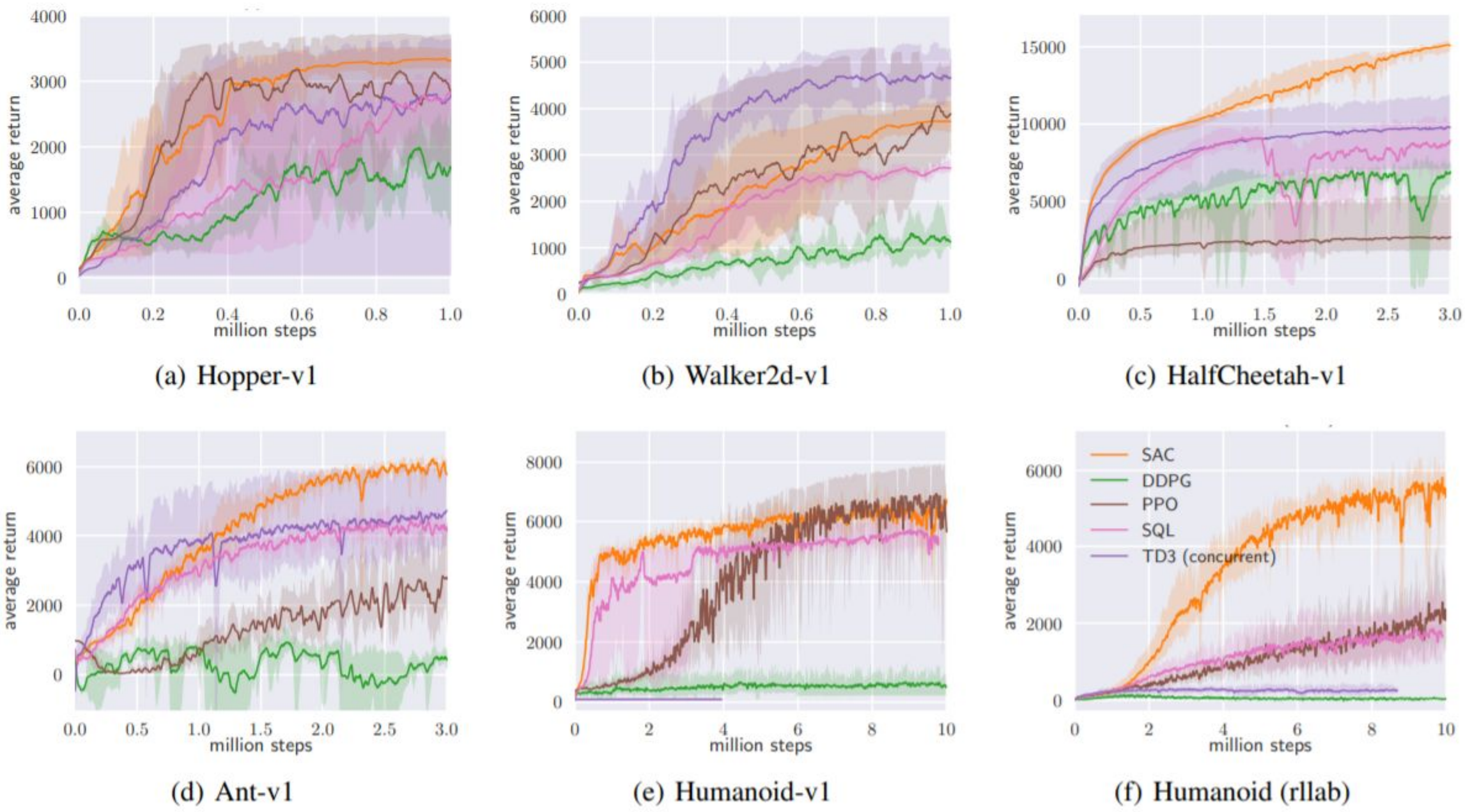
- Original paper learns V to stabilize training
- But in the second paper, V is not learned (reasons unclear)

Experimental Results

- Tasks
  - A range of continuous control tasks from the OpenAI gym benchmark suite
  - RL-Lab implementation of the Humanoid task
  - The easier tasks can be solved by a wide range of different algorithms, the more complex benchmarks, such as the 21-dimensional Humanoid (rllab) are exceptionally difficult to solve with off-policy algorithms.
- Baselines:
  - DDPG, SQL, PPO, TD3 (concurrent)
  - TD3 is an extension to DDPG that first applied the double Q-learning trick to continuous control along with other improvements.

https://arxiv.org/abs/1801.01290

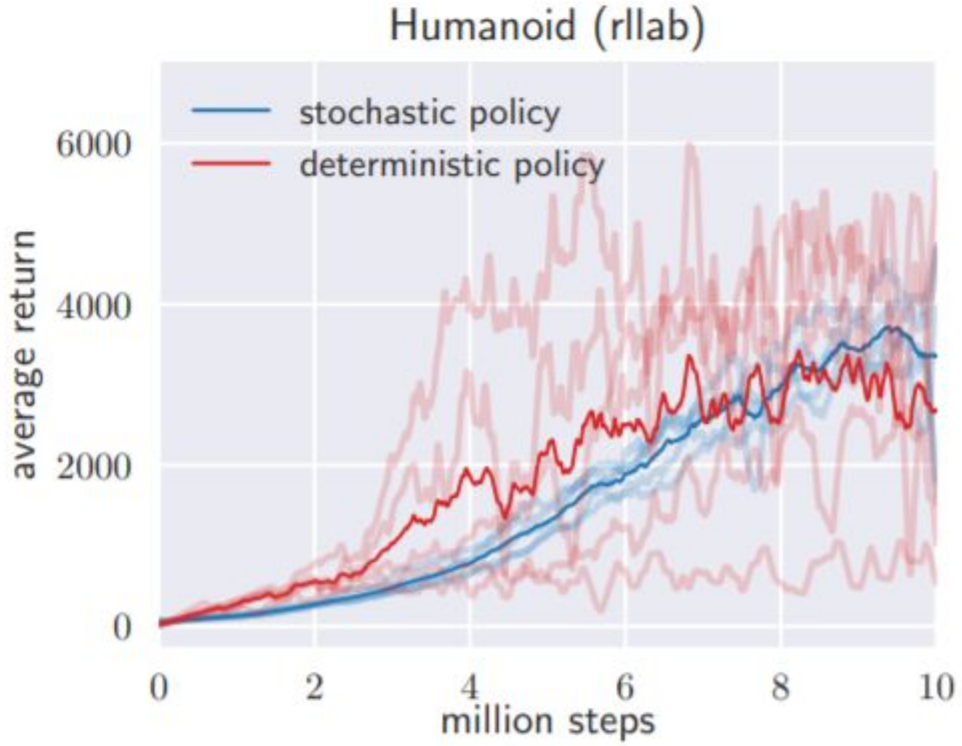
SAC: Results



average return
million steps
(a) Hopper-v1
(b) Walker2d-v1
(c) HalfCheetah-v1
(d) Ant-v1
(e) Humanoid-v1
(f) Humanoid (rllab)
SAC
DDPG
PPO
SQL
TD3 (concurrent)

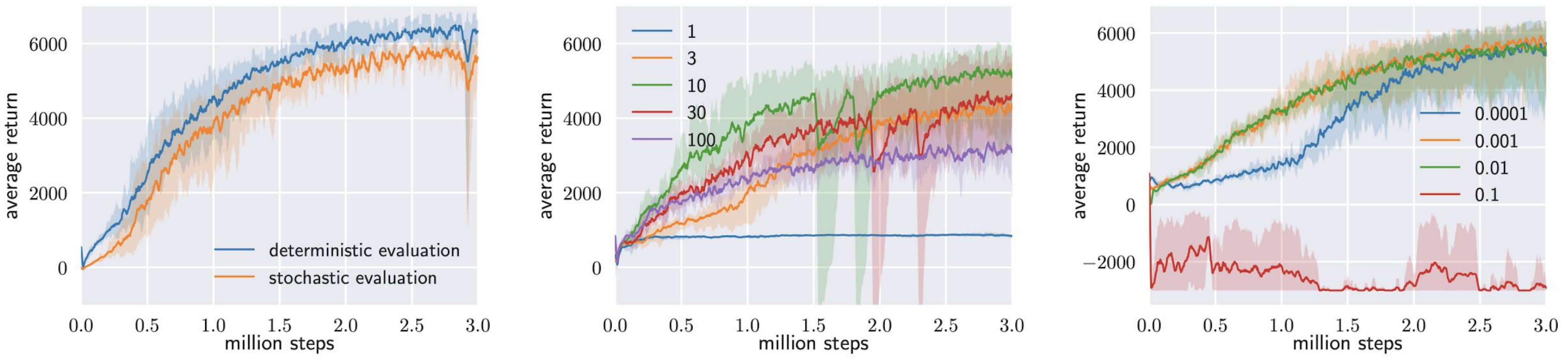
Experimental Results: Ablation Study

- How does the stochasticity of the policy and entropy maximization affect the performance?
- Comparison with a deterministic variant of SAC that does not maximize the entropy and that closely resembles DDPG



https://arxiv.org/abs/1801.01290

Experimental Results: Hyperparameter Sensitivity



(a) Evaluation (b) Reward Scale (c) Target Smoothing Coefficient (τ)

https://arxiv.org/abs/1801.01290

Limitation

- Unfortunately, SAC also suffers from brittleness to the alpha temperature hyperparameter that controls exploration
  - -> automatic temperature tuning!

Soft Actor-Critic Algorithms and Applications

Thomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Tikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, Sergey Levine

Contributions

- Adaptive temperature coefficient

temperature

$$\sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_t,\mathbf{a}_t) + \alpha\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t))\right]$$

- Extend to real-world tasks such as locomotion for a quadrupedal robot and robotic manipulation with a dexterous hand

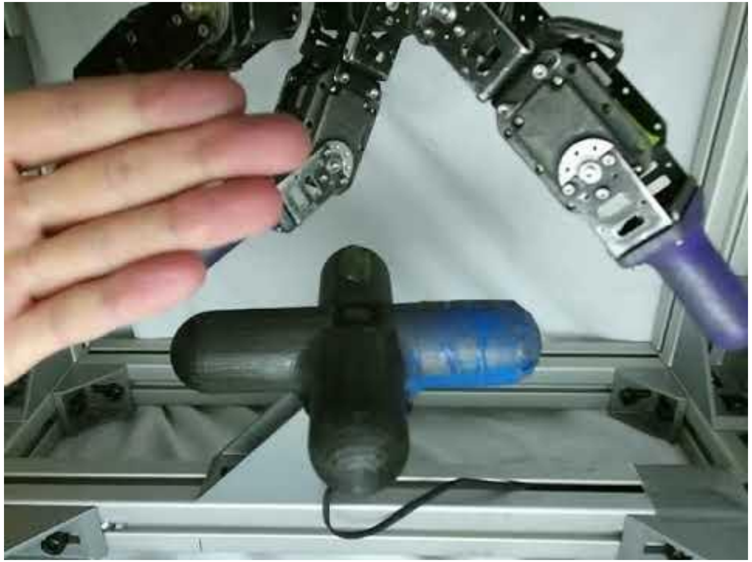
https://arxiv.org/abs/1801.01290

Real World Robots

https://arxiv.org/abs/1801.01290

Real World Robots

- Dexterous Hand Manipulations
- 20 hour end-to-end learning
- valve position as input: SAC 3 hours vs. PPO 7.4 hours



https://sites.google.com/view/sac-and-applications

Automatic Temperature Tuning

- Choosing the optimal temperature is non-trivial (tuned for each task)
- Constrained optimization problem:

$$\max \sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_t,\mathbf{a}_t) + \alpha\mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t))\right]$$

$$\max_{\pi_{0:T}} \mathbb{E}_{\rho_\pi}\left[\sum_{t=0}^{T} r(\mathbf{s}_t,\mathbf{a}_t)\right] \text{ s.t. } \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[-\log(\pi_t(\mathbf{a}_t|\mathbf{s}_t))\right] \geq \mathcal{H} \;\; \forall t$$

https://arxiv.org/abs/1801.01290

Dual Problem for the Constrained Optimization

Unroll the expectation

$$\max_{\pi_0}\left(\mathbb{E}\left[r(\mathbf{s}_0,\mathbf{a}_0)\right]+\max_{\pi_1}\left(\mathbb{E}\left[\ldots\right]+\boxed{\max_{\pi_T}\mathbb{E}\left[r(\mathbf{s}_T,\mathbf{a}_T)\right]}\right)\right)$$

For the last time step in the trajectory

$$\max_{\pi_T}\mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_T,\mathbf{a}_T)\right]=\min_{\alpha_T\geq 0}\max_{\pi_T}\mathbb{E}\left[r(\mathbf{s}_T,\mathbf{a}_T)-\alpha_T\log\pi(\mathbf{a}_T|\mathbf{s}_T)\right]-\alpha_T\mathcal{H}$$

$$\arg\min_{\alpha_T}\mathbb{E}_{\mathbf{s}_t,\mathbf{a}_t\sim\pi_t^*}\left[-\alpha_T\log\pi_T^*(\mathbf{a}_T|\mathbf{s}_T;\alpha_T)-\alpha_T\mathcal{H}\right].$$

Dual Problem for the Constrained Optimization

Similarly, for the previous time step

$$\max_{\pi_{T-1}} \left(\mathbb{E}\left[r(\mathbf{s}_{T-1}, \mathbf{a}_{T-1})\right] + \max_{\pi_T} \mathbb{E}\left[r(\mathbf{s}_T, \mathbf{a}_T)\right] \right) \tag{16}$$

$$= \max_{\pi_{T-1}} \left(Q^*_{T-1}(\mathbf{s}_{T-1}, \mathbf{a}_{T-1}) - \alpha_T \mathcal{H}\right)$$

$$= \min_{\alpha_{T-1} \geq 0} \max_{\pi_{T-1}} \left(\mathbb{E}\left[Q^*_{T-1}(\mathbf{s}_{T-1}, \mathbf{a}_{T-1})\right] - \mathbb{E}\left[\alpha_{T-1} \log \pi(\mathbf{a}_{T-1}|\mathbf{s}_{T-1})\right] - \alpha_{T-1}\mathcal{H} \right) + \alpha^*_T \mathcal{H}.$$

$$\alpha^*_t = \arg\min_{\alpha_t} \mathbb{E}_{\mathbf{a}_t \sim \pi^*_t}\left[-\alpha_t \log \pi^*_t(\mathbf{a}_t|\mathbf{s}_t; \alpha_t) - \alpha_t \bar{\mathcal{H}}\right]$$

Algorithm 1 Soft Actor-Critic

Input: θ_1, θ_2, ϕ two soft Q-functions ▷ Initial parameters

$\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$ ▷ Initialize target network weights

$\mathcal{D} \leftarrow \emptyset$ ▷ Initialize an empty replay pool

for each iteration **do**

  for each environment step **do**

    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$ ▷ Sample action from the policy

    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ ▷ Sample transition from the environment

    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$ ▷ Store the transition in the replay pool

  end for

  for each gradient step **do**

    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$ ▷ Update the Q-function parameters

    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ ▷ Update policy weights

    $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$ ▷ Adjust temperature

    $\bar{\theta}_i \leftarrow \tau \theta_i + (1 - \tau)\bar{\theta}_i$ for $i \in \{1, 2\}$ ▷ Update target network weights

  end for exponential moving average

end for

Output: θ_1, θ_2, ϕ ▷ Optimized parameters

Experimental Results: RL Lab

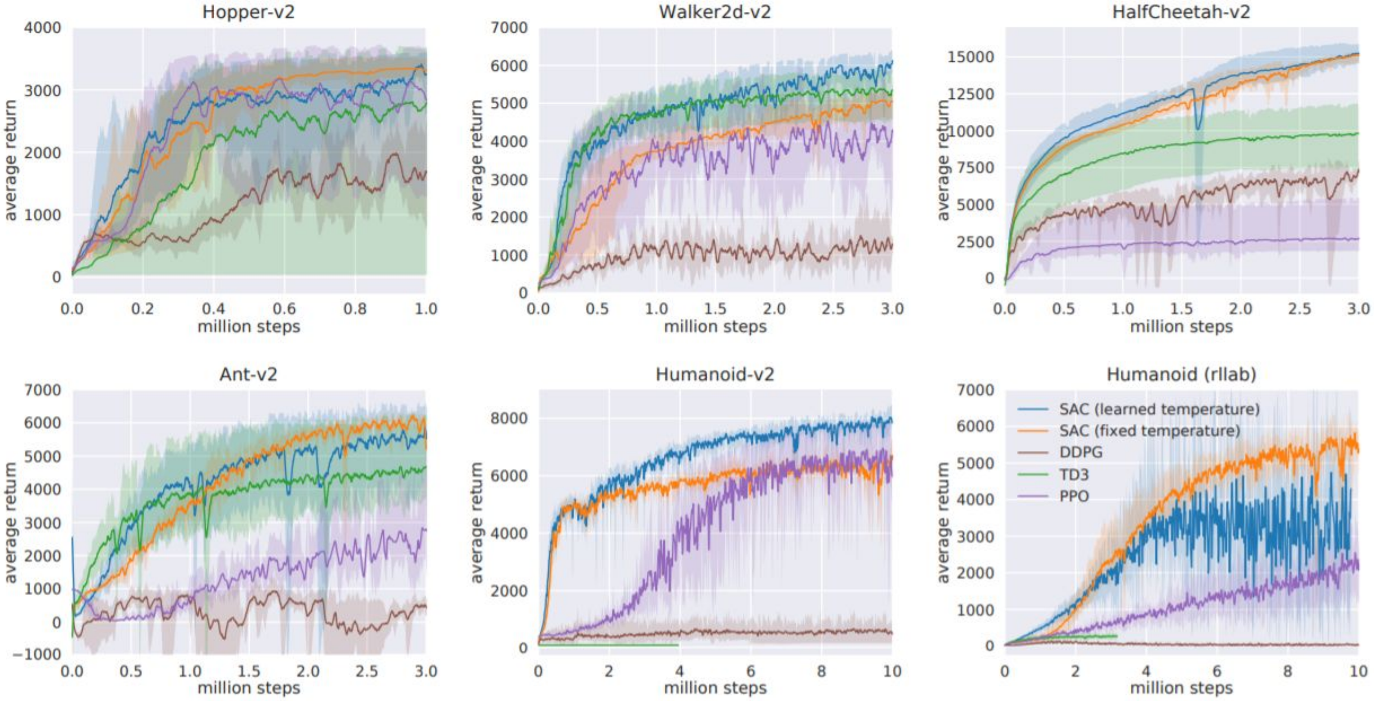
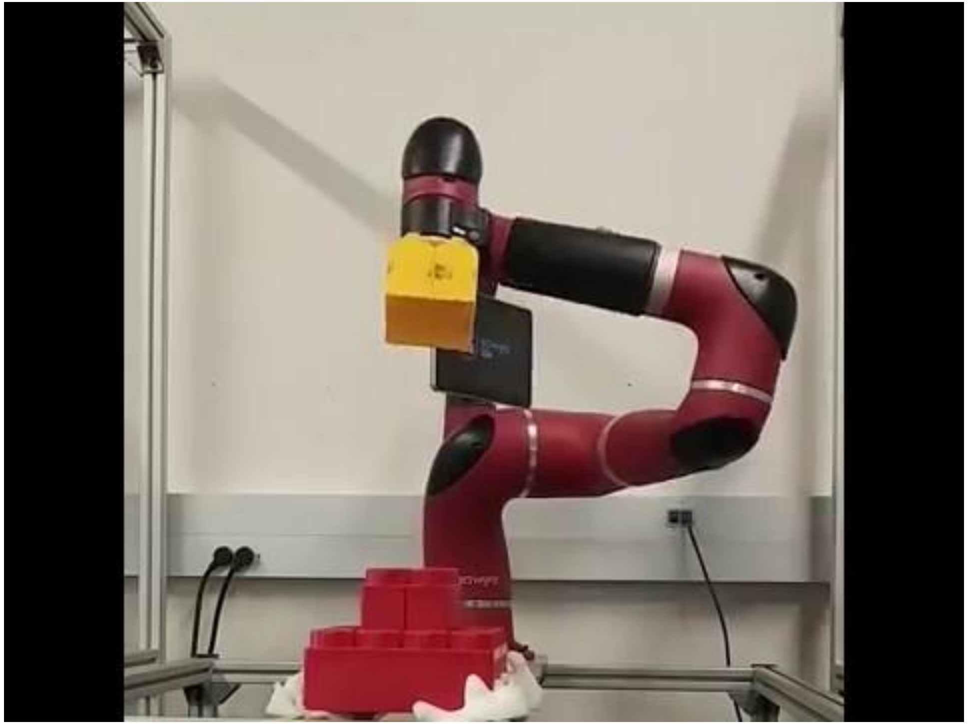


Figure 1: Training curves on continuous control benchmarks. Soft actor-critic (blue and yellow) performs consistently across all tasks and outperforming both on-policy and off-policy methods in the most challenging tasks.

https://arxiv.org/abs/1801.01290

Experimental Results: Robustness

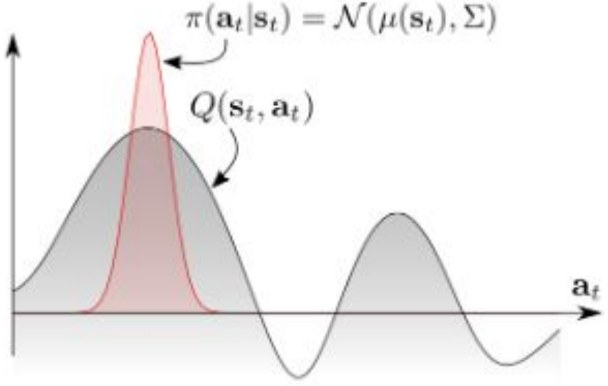


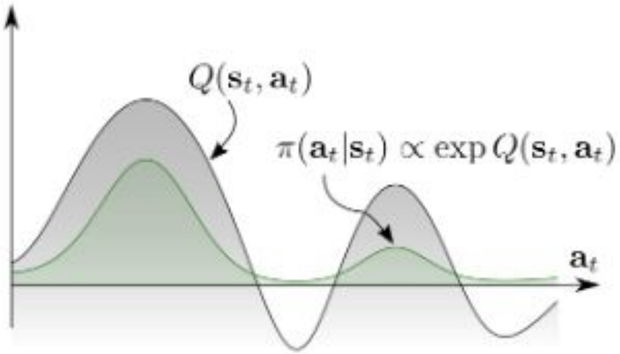
Limitations/Open Issues

- Lack of experiments on hard-exploration problem

Limitations/Open Issues

- Lack of experiments on hard-exploration problem
- Approximating a multi-modal Boltzmann distribution with a unimodal Gaussian



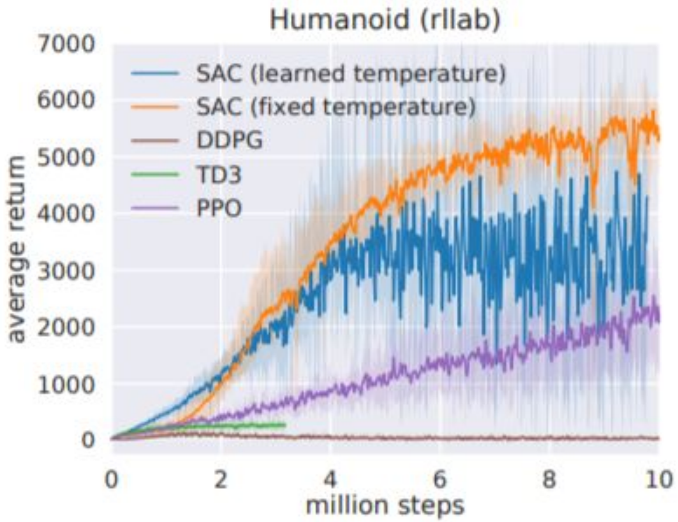


Limitations/Open Issues

- Lack of experiments on hard-exploration problem
- Approximating a multi-modal Boltzmann distribution with a unimodal Gaussian
- High-variance using automatic temperature tuning

Limitations/Open Issues

- Lack of experiments on hard-exploration problem
- Approximating a multi-modal Boltzmann distribution with a unimodal Gaussian
- High-variance using automatic temperature tuning



Recap: SAC

- An off-policy maximum entropy deep reinforcement learning algorithm
  - Sample-efficient
  - Scale to high-dimensional observation/action space
  - Robustness to random seed, noise and etc.
- Theoretical Results
  - Convergence of soft policy iteration
  - Derivation of soft-actor critic algorithm
- Empirical Results
  - SAC outperforms SOTA model-free deep RL methods, including DDPG, PPO and Soft Q-learning, in terms of the policy’s optimality, sample complexity and robustness.

Questions to test your understanding

- What is the objective in maximum entropy reinforcement learning?
- Why are off-policy methods more sample-efficient compared to on-policy methods?
- Why do we want the policy to be close to the exponential transformation of Q-value?
- What is soft policy iteration?

Any Questions?

Thank you!

Background: Q-Learning

- Q-Learning: use any behavioral policy to estimate the optimal Q* function that maximizes the future reward
  - Directly approximate Q* with Bellman Optimality Equation
  - Independent of policy being followed

Learning Rate Discount Factor

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \left(R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \right)$$

New State Old State Reward

https://www.youtube.com/watch?v=zR11FLZ-O9M

Max Entropy RL

- Entropy $$H(P) = \mathop{\mathrm{E}}_{x \sim P}\left[-\log P(x)\right]$$

- Entropy-regularized Reinforcement Learning

$$\pi^* = \arg\max_{\pi} \mathop{\mathrm{E}}_{\tau \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t \Big(R(s_t, a_t, s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right)\Big)\right]$$

- State Value Function & Value Function Q

$$V^{\pi}(s) = \mathop{\mathrm{E}}_{\tau \sim \pi}\left[\left.\sum_{t=0}^{\infty} \gamma^t \Big(R(s_t, a_t, s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right)\Big)\right| s_0 = s\right]$$

$$V^{\pi}(s) = \mathop{\mathrm{E}}_{a \sim \pi}\left[Q^{\pi}(s,a)\right] + \alpha H\left(\pi(\cdot|s)\right)$$

$$Q^{\pi}(s,a) = \mathop{\mathrm{E}}_{\tau \sim \pi}\left[\left.\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) + \alpha \sum_{t=1}^{\infty} \gamma^t H\left(\pi(\cdot|s_t)\right)\right| s_0 = s, a_0 = a\right]$$

https://spinningup.openai.com/en/latest/algorithms/sac.html

Real World Robots

- Need the ability to generalize to unseen environment and robustness against noisy real-world environment

- Robots get damaged in the physical world
  - requires sample-efficient learning
- Examples
  - Quadrupedal Locomotion in the Real World (2 hours of training)
  - Dexterous Hand Manipulations (20 hours end-to-end learning)

Real World Robots

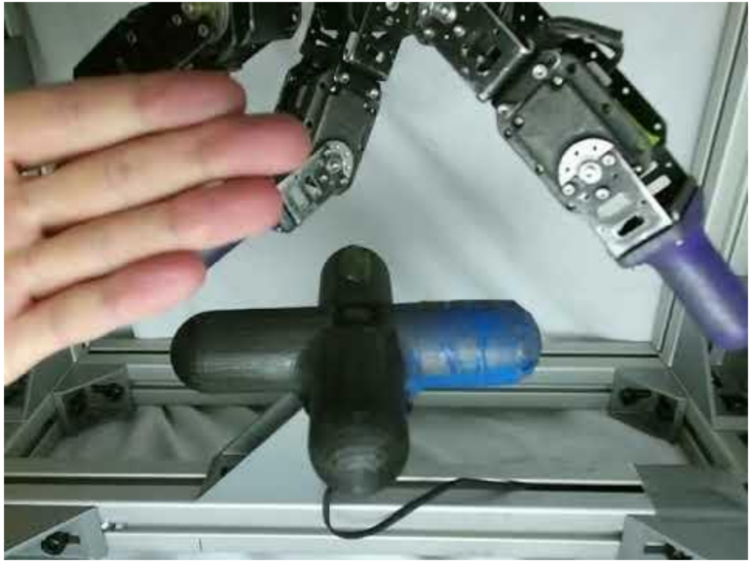
- Minitaur robot (Kenneally et al., 2016)

"first example of a DRL algorithm learning underactuated quadrupedal locomotion directly in the real world without any simulation or pretraining"

https://sites.google.com/view/sac-and-applications

Real World Robots

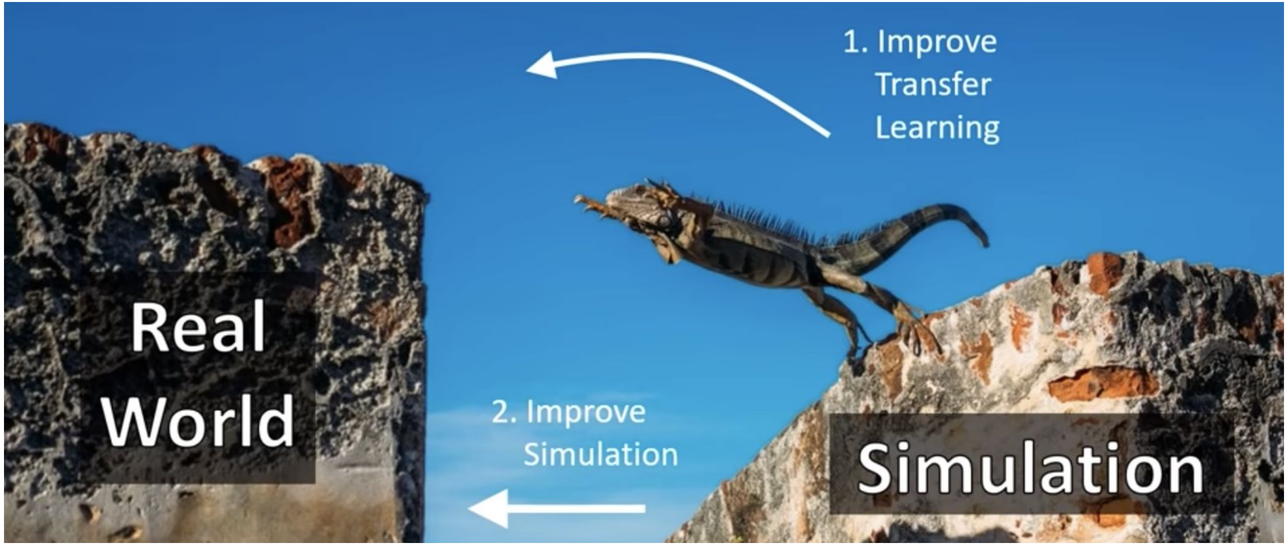
- Dexterous Hand Manipulations
- 20 hour end-to-end learning
- valve position as input: SAC 3 hours vs. PPO 7.4 hours



https://sites.google.com/view/sac-and-applications

Main Problem - Sample Inefficiency

- Sample inefficient algorithms can be problematic when deployed in the real world
    - damage to robots/humans



Main Problems

widespread adoption of model-free DRL is hampered by:

- expensive in terms of sample complexity
  - simple tasks require millions of steps of data collection
  - high-dimensional observations/action space require substantially more
- brittle with respect to hyperparameters
  - learning rates, exploration constants
  - set carefully to achieve good results

Soft actor-critic

1. Q-function update

Update Q-function to evaluate current policy:

$$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \mathbb{E}_{\mathbf{s}'\sim p_{\mathbf{s}},\ \mathbf{a}'\sim\pi}\left[Q(\mathbf{s}',\mathbf{a}') - \log\pi(\mathbf{a}'|\mathbf{s}')\right]$$

This converges to Q^π.

2. Update policy

Update the policy with gradient of information projection:

$$\pi_{\text{new}} = \arg\min_{\pi'} \mathrm{D}_{\mathrm{KL}}\left(\pi'(\,\cdot\,|\mathbf{s}) \,\middle\|\, \frac{1}{Z}\exp Q^{\pi_{\text{old}}}(\mathbf{s},\,\cdot\,)\right)$$

In practice, only take one gradient step on this objective

3. Interact with the world, collect more data

Haarnoja, Zhou, Abbeel, L., *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning*. 2018