Soft Actor-Critic

Review and application of Maximum Entropy Reinforcement Learning

Isabella Carla Gonnella **Dario Coscia** **Alessandro Pierro**

Reinforcement Learning
MSc in Data Science and Scientific Computing

University of Trieste

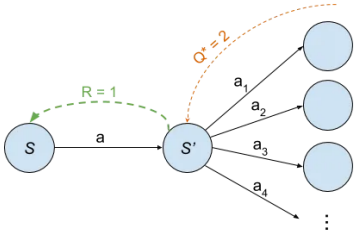
Aim of the Project

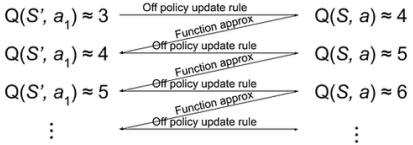
- The aim of this project is to present the **Soft Actor Critic (SAC)** algorithm, reproducing the results from the original paper
- We implemented the algorithm and tested it on continuous control tasks from the Mujoco simulator
- The code is available at: https://github.com/DSSC-projects/soft-actor-critic

Table of Contents

The Deadly Triad

- Off-policy
- Bootstrapping
- Function Approximation





$$Q(s,a) = Q(s,a) + \alpha(R + \gamma \arg\max_a Q(s',a) - Q(s,a))$$

Maximum Entropy Reinforcement Learning

- Introduced in [5]
- The standard reinforcement learning objective is augmented with a term proportional to the **entropy of the policy** $\mathcal{H}(\pi(\cdot|\mathbf{s}_t))$
- The optimal policy is then defined as:

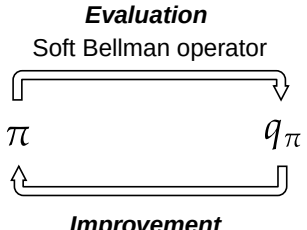
$$\pi^* = \arg\max_{\pi} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \pi} [r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot|\mathbf{s}_t))]$$

- Since the entropy term controls the stochasticity of the policy, this formulation provides a natural framework to **balance exploration and exploitation** in large action spaces

Soft Actor-Critic (SAC)

Soft Actor-Critic

- Soft Actor-Critic (SAC) [2] is an **off-policy** actor-critic deep RL algorithm based on the **maximum entropy** framework
- The actor objectives is to maximize expected reward while also maximizing entropy, i.e acting as randomly as possible.
- SAC uses neural networks to approximate the policy π, the state-value function v_π, and the state-action value function q_π.



Evaluation phase

Given the soft state-value function

$$v_\pi(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t)}[q_\pi(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi(\mathbf{a}_t \mid \mathbf{s}_t)]$$

and the soft Bellmann Operator

$$\mathcal{B}^\pi q_\pi(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p(\cdot|\mathbf{s}_t, \mathbf{a}_t)}[v_\pi(\mathbf{s}_{t+1})]$$

the **evaluation phase** consist of iteratively applied $\mathcal{B}^\pi$ to q_π.

Lemma 1 - Soft Policy Evalutation

Consider the soft Bellman operator $\mathcal{B}^\pi$, and a mapping $q_\pi^0 : S \times A \to \mathbb{R}$, with $|A| < \infty$, and define $q_\pi^{k+1} = \mathcal{B}^\pi q_\pi^k$. Then the sequence q_π^k will converge to the soft Q-value of π as $k \to \infty$.

Improvement phase

The **improvement phase** consist in projecting the policy π in a set of tractable policies Π, using Kullback-Leibler divergence as information projection

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} D_{\text{KL}} \left(\pi' \| \frac{\exp(q_{\pi_{\text{old}}})}{Z_{\pi_{\text{old}}}} \right)$$

The partition function $Z_{\pi_{\text{old}}}$ normalizing the distribution does not contribute to the gradient with respect to the new policy and can thus be ignored

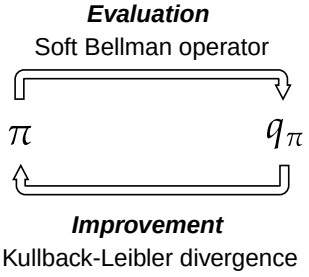
Lemma 2 - Soft Policy Improvement

Let $\pi_{\text{old}} \in \Pi$ and let $\pi_{\text{new}} \in \Pi$ be the optimizer of the minimization problem defined in terms of Kullback-Leibler divergence. Then $q_{\pi_{\text{new}}}(\mathbf{s}, \mathbf{a}) \geq q_{\pi_{\text{old}}}(\mathbf{s}, \mathbf{a})$ for all $(\mathbf{s}, \mathbf{a}) \in S \times A$ with $|A| < \infty$.

Soft Policy Iteration theorem

Theorem 1 - Policy Iteration

Repeated application of soft policy evaluation and soft policy improvement from any $\pi \in \Pi$ converges to a policy π^* such that $q_\pi^*(\mathbf{s}, \mathbf{a}) \geq q_\pi(\mathbf{s}, \mathbf{a})$ for all $\pi \in \Pi$ and $(\mathbf{s}, \mathbf{a}) \in S \times A$, assuming $|A| < \infty$.



Soft Actor Critic Algorithm

The SAC algorithm uses multiple neural networks:

- $v_\pi(\mathbf{s}; \psi) \rightarrow$ approximation soft value function
- $v_\pi(\mathbf{s}; \bar{\psi}) \rightarrow$ target soft value function using exponential moving average
- $q_\pi(\mathbf{s}, \mathbf{a}; \theta_i) \rightarrow$ approximation soft Q-value functions (critics)
- $\pi(\mathbf{a} \mid \mathbf{s}; \phi) \rightarrow$ approximation policy (actor)

With the following objective functions:

$$J_v(\psi) = \mathbb{E}_{\mathbf{s}_t \sim D}\left[1/2(v_\pi(\mathbf{s}; \psi) - \mathbb{E}_{\mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t;\phi)}[q_\pi(\mathbf{s}_t, \mathbf{a}_t; \theta) - \alpha \log \pi(\mathbf{a}_t \mid \mathbf{s}_t; \phi)])^2\right]$$

$$J_q(\theta) = \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t \sim D}\left[1/2(q_\pi(\mathbf{s}, \mathbf{a}; \theta) - r(\mathbf{s}_t, \mathbf{a}_t) - \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p(\cdot|\mathbf{s}_t, \mathbf{a}_t)}[v_\pi(\mathbf{s}_{t+1}; \bar{\psi})])^2\right]$$

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

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

Algorithm details: Reparametrization trick

The gradient of the policy loss $\nabla J_\pi(\phi)$ can be estimated in a similar fashion as done in REINFORCE algorithm:

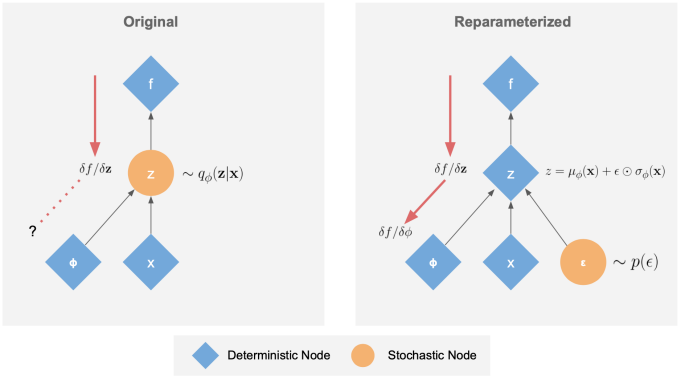
$$\nabla J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim D, \mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t;\phi)} \left[(\alpha \log \pi(\mathbf{a}_t|\mathbf{s}_t;\phi) - q_\pi(\mathbf{s}_t, \mathbf{a}_t;\theta)) \nabla \log \pi(\mathbf{a}_t|\mathbf{s}_t;\phi) \right]$$

However, this estimate is usually characterized by a higher variance!

The reparameterization trick [4] is a useful technique to rewrite $\nabla J_\pi(\phi)$ as a differentiable Monte Carlo estimate w.r.t. ϕ, which has been shown to obtain samples with lower variance

Algorithm details: Reparametrization trick

With the reparametrization trick we rewrite $\mathbf{a}_t$ as a deterministic function of ϕ, by moving the stochasticity to a variable $\epsilon \sim p(\epsilon)$



Algorithm details: Double Q-value

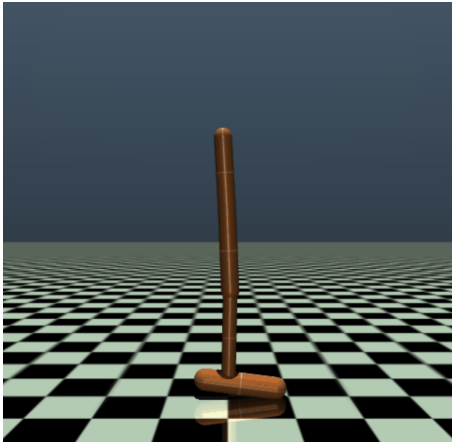
- As noted in [3], the q-value estimate introduces a positive bias, assuming the greedy action will always be selected
- SAC adopts the Double Q-value trick, maintaining two critics $q_\pi(\mathbf{s}, \mathbf{a}; \theta_1), q_\pi(\mathbf{s}, \mathbf{a}; \theta_2)$ that are trained independently
- The minimum across the two Q-values is used in gradient computation, thus reducing the overestimation
- Another reason for using an ensemble of q-function is to avoid overfitting the policy to a sub-optimal q-function during training

Algorithm details: Action scaling

- A Gaussian distribution $\mu(\mathbf{z} \mid \mathbf{s})$ is used for explicitly parametrized the policy, with $z \in \mathbb{R}^D$.
- To bound the action samples an invertible squashing function ($\tanh$) to the Gaussian samples is used, $\mathbf{a} = \tanh(\mathbf{z})$
- The log-probability distribution can be the change of variable formula $\log \pi(\mathbf{a} \mid \mathbf{s}) = \log \mu(\mathbf{z} \mid \mathbf{s}) - \sum_{i=1}^{D} \log(1 - \tanh(z_i)^2)$

Applications

Task A: Hopper



Action Space → $dim = 3$

Action	min	max
3 *torques applied on different points*	-1.0	1.0

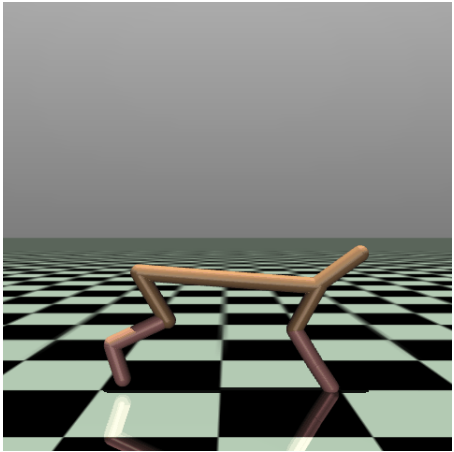
Observation Space → $dim = 11$

Observation	min	max
11 *angles, velocity and position of different points*	$-\infty$	∞

Reward

$r = r_f + r_h - c_{ctrl}$

Task B: HalfCheetah



Action Space → $dim = 6$

Action	min	max
6 *torques applied on different points*	-1.0	1.0

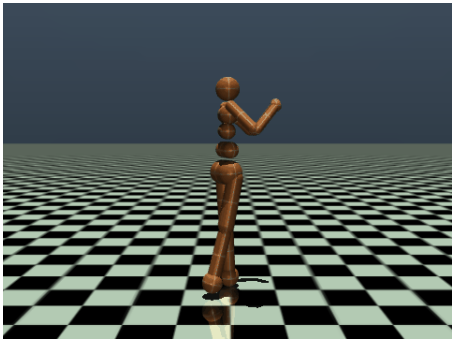
Observation Space → $dim = 17$

Observation	min	max
17 *angles, velocity and position of different points*	$-\infty$	∞

Reward

$r = r_f - c_{ctrl}$

Task C: Humanoid



Action Space → $dim = 17$

Action	min	max
17 *torques applied on different points*	-0.4	0.4

Observation Space → $dim = 45$

Observation	min	max
45 *angles, velocity and position of different points*	$-\infty$	∞

Reward

$r = r_f + r_h - c_{ctrl} - c_{co}$

Experimental Results

Experiment setup

For the experiments we built the same networks as suggested in [1]. For all tests we used $\alpha = 0.2$, expect for the Humanoid where we used $\alpha = 0.5$.

Parameter	Value
Shared	
optimizer	Adam
learning rate	$3 \cdot 10^{-4}$
discount (γ)	0.99
replay buffer size	10^6
number of hidden layers (all networks)	2
number of hidden units per layer	256
number of samples per minibatch	256
nonlinearity	ReLU
SAC	
target smoothing coefficient (τ)	0.005
target update interval	1
gradient steps	1

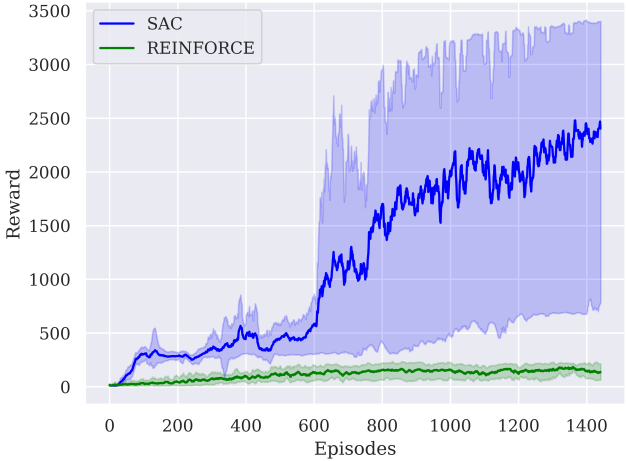


Figure 1: `Hopper-v4`: total episodic reward during training across 3 different runs. The solid line represents the mean, and the shaded area the min/max range.

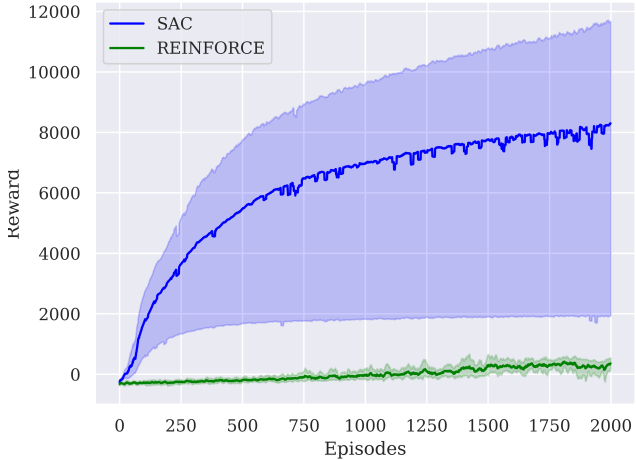


Figure 2: `HalfCheetah-v4`: total episodic reward during training across 3 different runs. The solid line represents the mean, and the shaded area the min/max range.

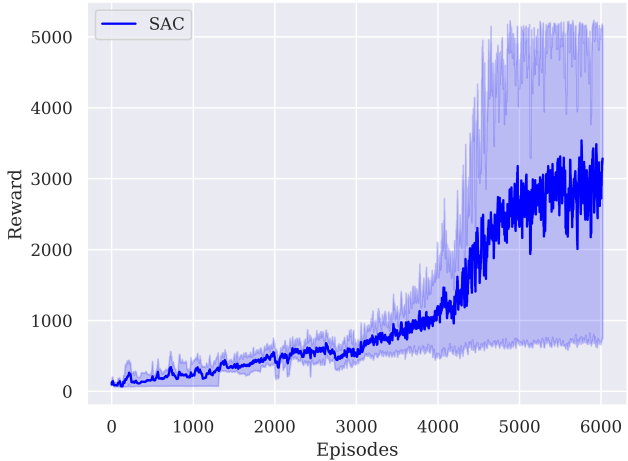


Figure 3: `Humanoid-v4`: total episodic reward during training across 3 different runs. The solid line represents the mean, and the shaded area the min/max range.

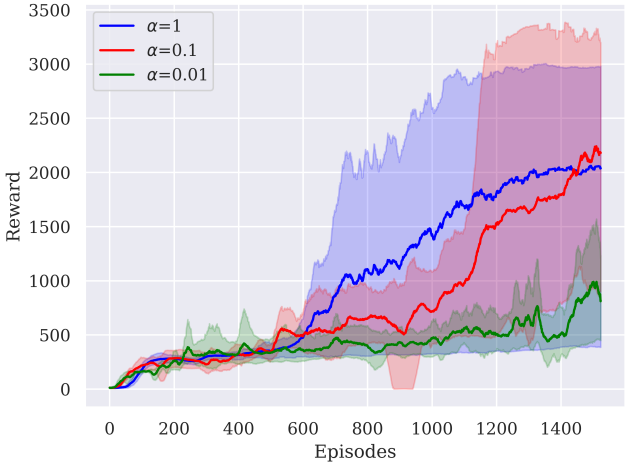


Figure 4: `Hopper-v4`: total episodic reward during training across 3 different runs for 3 different values of α. The solid line represents the mean, and the shaded area the min/max range.

Conclusions

Conclusions

- We presented the **Soft Actor-Critic method (SAC)**, reproducing the main results of the original paper [2].
- We chose the **REINFORCE algorithm as baseline** and compared the results on different test cases.
- Overall, SAC performs significantly better than REINFORCE
- We performed an ablation study, evaluating the impact of the **temperature parameter** α on the training.

Further literature

- One of the main criticality of SAC is the fact that the temperature α is kept fix during the training, meaning that exploration never ends
- In [2] the authors propose to learn the optimal value of α by recasting the learning procedure as a contraint problem by fixing a lower bound on the entropy
- Another problem is related to sampling from the buffer, where past experience is sampled uniformally
- A different prioritized sampling strategy is proposed in [1], where the buffer favours samples with higher episodic returns. Additionally the authors include the latest on-policy experiences for performing the gradient update

Contributions

- Literature review: **everyone**
- SAC implementation: **Dario Coscia**
- Experiments: **Isabella Carla Gonnella, Alessandro Pierro**
- Presentation: **everyone**

References

[1] Chayan Banerjee, Zhiyong Chen, and Nasimul Noman. “Improved soft actor-critic: Mixing prioritized off-policy samples with on-policy experiences”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[2] Tuomas Haarnoja et al. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. In: *International conference on machine learning*. PMLR. 2018.

[3] Hado Hasselt. “Double Q-learning”. In: *Advances in Neural Information Processing Systems*. Vol. 23. 2010.

[4] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[5] Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

Thank you for the attention!

Isabella Carla Gonnella, Dario Coscia, Alessandro Pierro