Reinforcement Learning

A study based on Soft Actor-Critic

Lucien Le Gall, Alexandre Zenou, Remi Moshfeghi

Ecole Polytechnique : M2DS

January 2026

Introduction: Deep Reinforcement Learning for Continuous Control

- **Model-free deep reinforcement learning** aims to learn a policy $\pi(a \mid s)$ from interactions, **without knowing** the environment dynamics.
- This framework has shown strong results on complex tasks:
  - games (Taxi-v3, Go),
  - robotic control / locomotion (e.g., MuJoCo).
- However, its use in real-world settings remains limited:

$$\text{high data cost} \quad + \quad \text{training instability.}$$

Two major challenges in model-free deep RL

(1) Very high sample complexity

Even simple tasks may require **millions of samples** to learn, and high-dimensional tasks require even more.

(2) Brittle convergence

Performance strongly depends on hyperparameters:
$\Rightarrow$ careful tuning is required to obtain good results.

- These two issues strongly limit the applicability of deep RL methods to real tasks.

Why these difficulties? On-policy vs Off-policy

On-policy $\Rightarrow$ data inefficiency

Many popular methods (TRPO, PPO, A3C) are **on-policy**:

- they require collecting **new trajectories** for each update,
- which becomes extremely costly when the task is complex.

Off-policy $\Rightarrow$ better sample efficiency

Off-policy methods reuse past experience (replay buffer), but they often become harder to stabilize.

Off-policy learning + function approximation: a stability challenge

- In deep RL, the combination:

  off-policy learning + neural networks + continuous spaces

  raises **stability** and **convergence** issues.
- In continuous action spaces, a maximization $\max_a Q(s, a)$ is non-trivial: $\Rightarrow$ one often introduces an **actor** network in addition to the critic.

Example: DDPG

- **sample-efficient** (off-policy),
- but **very brittle** and extremely sensitive to hyperparameters.

Preliminaries: Infinite-horizon MDP (continuous control)

- We consider an infinite-horizon MDP:

$$(\mathcal{S}, \mathcal{A}, p, r)$$

  with $\mathcal{S}$ and $\mathcal{A}$ **continuous**.
- $p(s_{t+1} \mid s_t, a_t)$: unknown transition density.
- $r(s_t, a_t) \in [r_{\min}, r_{\max}]$: bounded reward.
- A policy $\pi(a \mid s)$ induces a trajectory distribution, with marginals:

$$\rho_\pi(s_t) \quad \text{and} \quad \rho_\pi(s_t, a_t).$$

Maximum Entropy Reinforcement Learning

- In standard RL, we only maximize the expected reward.
- Here, we adopt the **maximum entropy** framework: we encourage stochastic policies by adding an entropy regularization term.

Objective (entropy-regularized)

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t, a_t)\sim\rho_\pi} \Big[r(s_t, a_t) + \alpha\, \mathcal{H}(\pi(\cdot \mid s_t)) \Big].$$

with $\mathcal{H}(\pi(\cdot \mid s_t)) = -\mathbb{E}_{a_t\sim\pi}[\log \pi(a_t \mid s_t)]$

- $\alpha > 0$: temperature (controls the reward/exploration trade-off)
- When $\alpha \to 0$, we recover the standard RL objective

Why add entropy? (intuition)

- **Improved exploration**: the agent avoids becoming deterministic too early.
- **Multi-modality**: if several behaviors are near-optimal, the policy can represent them.
- **Robustness**: maximum entropy policies are more robust to estimation errors.

Simple interpretation

Solve the task (reward) $+$ stay random (entropy).

Towards Soft Actor-Critic: why start with Soft Policy Iteration?

- Historically, maximum entropy methods were often formulated via **soft Q-learning**, which leads to difficulties in continuous action spaces (complex approximate inference).
- The goal of the paper is to build a method that is:
  - **off-policy** (sample efficiency),
  - **actor-critic** (suitable for continuous control),
  - **maximum entropy** (stability + exploration).

Outline

- The **Soft Actor-Critic (off-policy)** algorithm is derived from a **maximum entropy variant of policy iteration**.
- We will first introduce a practical Soft Policy Iteration algorithm, before presenting Soft Actor-Critic as described in the paper under study.

Soft Policy Iteration: general idea

- Goal: learn an optimal **maximum entropy** policy through a variant of **policy iteration**.
- The algorithm alternates between two steps:
  1. **Policy Evaluation**: evaluate policy π (soft Q^{π})
  2. **Policy Improvement**: improve π (update towards a soft-optimal policy)
- The derivation is first done in the **tabular** setting ($|\mathcal{A}| < \infty$) $\Rightarrow$ theoretical analysis + convergence guarantees.

Goal of this section

Show that **Soft Policy Iteration** converges to the best policy within a class Π (e.g., a family of parameterized densities).

Soft Policy Evaluation: modified Bellman backup

- For a fixed policy π, we aim to compute its **soft Q-value**.
- We introduce a **soft** Bellman operator:

Soft Bellman backup operator

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

- where the **soft state-value function** is:

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

Ref.: Eq. (2) and (3), Haarnoja et al. (2018).

Lemma 1: convergence of Soft Policy Evaluation

Lemma 1 (Soft Policy Evaluation)

Consider the operator $\mathcal{T}^\pi$ and an initial function $Q_0 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ with $|\mathcal{A}| < \infty$. Define

$$Q_{k+1} = \mathcal{T}^\pi Q_k,$$

then the sequence (Q_k) converges to the **soft Q-value** of π as $k \to \infty$.

- Interpretation: analogous to classical policy evaluation, but with an **entropy bonus built-in**.
- **Theoretical guarantees** hold thanks to the tabular setting.

Proof: Appendix B.1, Haarnoja et al. (2018).

Soft Policy Improvement: policy update

- After evaluation, we want a policy that moves closer to:

$$\pi(\cdot \mid s) \propto \exp(Q^{\pi_{\text{old}}}(s, \cdot)).$$

- In practice, we restrict the policy to a tractable class Π (e.g., parameterized densities: Gaussians).

Information projection (KL) in Π

- $Z^{\pi_{\text{old}}}(s_t)$ normalizes the distribution (partition function).

Soft Policy Improvement: policy update 2

Proposition

If a policy $\pi^{\text{new}}(\cdot|s)$ fulfills (5), and Π is the set of distributions over $\mathcal{A}$, then:

$$\pi_{\text{new}}(a|s) = \frac{\exp\left(q^{\pi^{\text{old}}}(s,a)\right)}{\sum_{a\in\mathcal{A}} \exp\left(q^{\pi^{\text{old}}}(s,a)\right)}. \tag{6}$$

Proof: See Appendix A (page 23).

Lemma 2: Soft Policy Improvement ⇒ guaranteed improvement

Lemma 2 (Soft Policy Improvement)

Let $\pi_{\text{old}} \in \Pi$ and let π_{new} be the solution of the KL problem above. Then:

$$Q^{\pi_{\text{new}}}(s_t, a_t) \;\geq\; Q^{\pi_{\text{old}}}(s_t, a_t), \qquad \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A},$$

(assuming $|\mathcal{A}| < \infty$).

- Interpretation: this update is a valid **policy improvement step** for the maximum entropy objective.

Proof: Appendix B.2, Haarnoja et al. (2018).

Soft Policy Iteration: convergence to the optimum in Π

- Soft Policy Iteration alternates between:
  1. Soft policy evaluation (Lemma 1)
  2. Soft policy improvement (Lemma 2)

Theorem 1 (Soft Policy Iteration)

Repeated application of these two steps converges to a policy $\pi^\star \in \Pi$ such that:

$$Q^{\pi^\star}(s_t, a_t) \;\geq\; Q^{\pi}(s_t, a_t), \qquad \forall \pi \in \Pi,\ \forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A},$$

(assuming $|\mathcal{A}| < \infty$).

Proof: Appendix B.3, Haarnoja et al. (2018).

From Soft Policy Iteration to Soft Actor-Critic

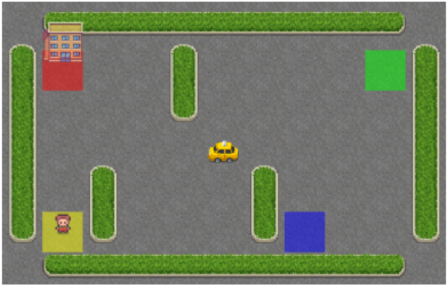
- In theory, SPI converges in the **tabular** case.
- But in continuous control (large spaces):
  - Q must be represented by a **function approximator** (neural networks),
  - running *evaluation/improvement until convergence* is too expensive.
- We therefore build a practical approximation:

Idea

Soft Actor-Critic = deep RL version of SPI, trained **off-policy**.

Illustration: Soft Policy Iteration

Illustration of Soft Policy Iteration with `env = gym.make("Taxi-v3")` – see GitHub



4.2 Soft Actor-Critic: core idea

- In continuous domains, we cannot run **Soft Policy Iteration** exactly.
- SAC is a **practical approximation**:
  - we approximate the **Q-function** and the **policy** with neural networks,
  - instead of running *evaluation/improvement* until convergence, we **alternate** updates using **stochastic gradient descent**.
- We consider three parameterized objects:

$$V_\psi(s_t), \qquad Q_\theta(s_t, a_t), \qquad \pi_\phi(a_t \mid s_t).$$

- Data is collected from a **replay buffer** $\mathcal{D}$ (off-policy).

Soft Actor-Critic: objective functions (Eq. 5–10)

(1) Soft Value Function V_ψ (Eq. 5)

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

(2) Soft Q-function Q_θ (Eq. 7–8)

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\frac{1}{2}\Big(Q_\theta(s_t, a_t) - \widehat{Q}(s_t, a_t)\Big)^2\right],$$

$$\widehat{Q} = r(s_t, a_t) + \gamma\, V_{\bar{\psi}}(s_{t+1}).$$

(3) Policy π_ϕ : KL minimization (Eq. 10)

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

Soft Actor-Critic: algorithm structure (Algorithm)

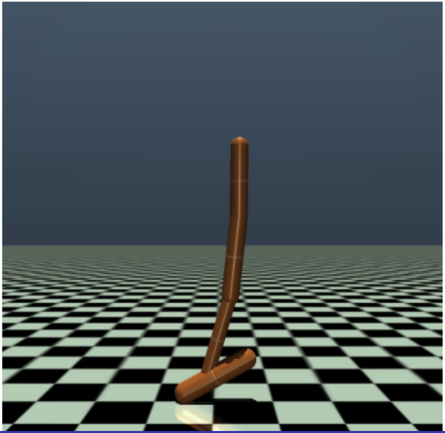
- **Initialize**: parameters $\psi, \theta_1, \theta_2, \phi$ and replay buffer $\mathcal{D}$.
- **Environment step**:

$$a_t \sim \pi_\phi(\cdot \mid s_t), \quad s_{t+1} \sim p(\cdot \mid s_t, a_t), \quad (s_t, a_t, r_t, s_{t+1}) \in \mathcal{D}.$$

- **Gradient step (batch from $\mathcal{D}$)**:
  - update V_ψ by minimizing $J_V(\psi)$
  - update $Q_{\theta_1}, Q_{\theta_2}$ by minimizing $J_Q(\theta_i)$
  - update π_ϕ by minimizing $J_\pi(\phi)$
  - target network: $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$
- Two important practical points:
  - **reparameterization trick**: $a_t = f_\phi(\varepsilon_t; s_t)$ (lower variance)
  - **double Q**: use $\min(Q_{\theta_1}, Q_{\theta_2})$ to reduce positive bias

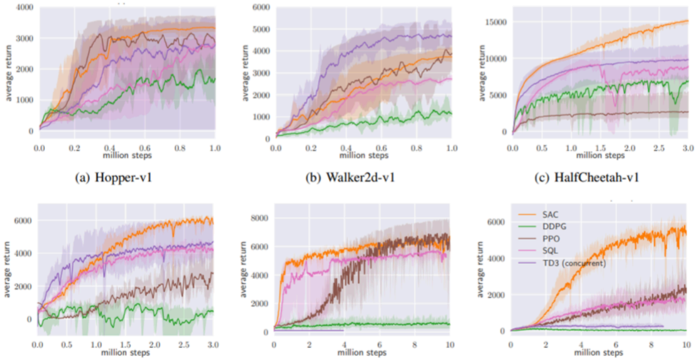
Illustration: Soft Actor-Critic

Illustration of Soft Actor-Critic with `env = gym.make("Hopper-v5")` – see GitHub



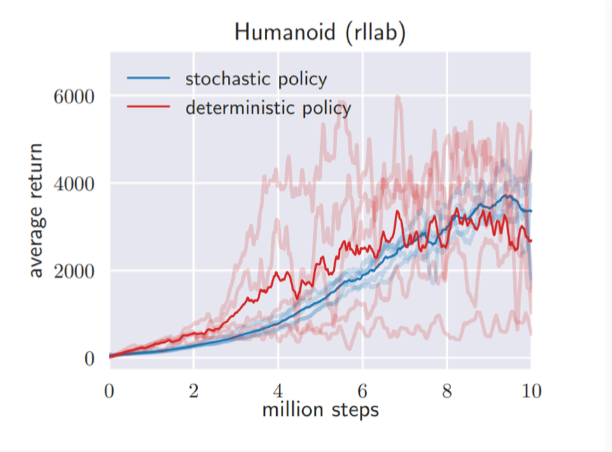
1. Introduction
2. Soft policy iteration
3. Soft actor critic
4. **Discussion**

Discussion —Algorithm comparison (MuJoCo)

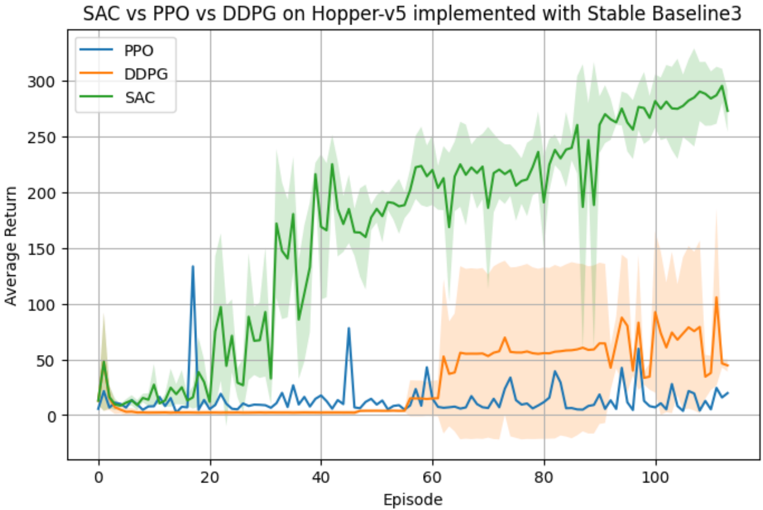


(a) Hopper-v1
(b) Walker2d-v1
(c) HalfCheetah-v1
(d) Ant-v1
(e) Humanoid-v1
(f) Humanoid (rllab)

Discussion —SAC: stochastic vs deterministic



Discussion —Personal results implemented with the Stable Baselines3 library



Thanks!