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

Dimitris Roidis   Nikolas Kavaklis

June 23, 2025

Contents

Motivation

- Traditional reinforcement learning algorithms suffer from:
  - **Instability during training** (e.g., DDPG)
  - **Overestimation bias** in Q-value learning
  - **Poor exploration** in high-dimensional action spaces
- **SAC addresses these issues by combining:**
  - Off-policy learning $\rightarrow$ *high sample efficiency*
  - Stochastic actor $\rightarrow$ *inherent exploration*
  - Entropy maximization $\rightarrow$ *robust and diverse behavior*
  - Double Q-learning and target networks $\rightarrow$ *reduces value overestimation*
- **Ideal for continuous action spaces and real-world robotics applications**.

Markov Decision Process (MDP)

An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$:

- $\mathcal{S}$: State space
- $\mathcal{A}$: Action space (continuous in SAC)
- $p(s'|s, a)$: Transition probability (next state given current state and action)
- $r(s, a)$: Reward signal from the environment
- γ: Discount factor, balancing immediate vs future rewards

Objective: Learn a policy $\pi(a|s)$ to maximize expected cumulative reward over time.

Off-Policy Learning and Maximum Entropy RL

- **Off-policy RL:** Learns from experience (s, a, r, s') stored in a replay buffer
- **Maximum entropy objective:**

$$J(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\sum_t r(s_t, a_t) + \alpha\,\mathcal{H}(\pi(a_t \mid s_t))\right]$$

- $\mathcal{H}(\pi(\cdot|s)) = -\mathbb{E}_{a\sim\pi}[\log \pi(a|s)]$: encourages stochastic, exploratory policies
- α: temperature parameter – balances reward vs entropy

Soft Bellman Backup Operator

The soft Bellman backup replaces the standard one:

$$Q(s,a) \leftarrow r + \gamma \cdot (Q(s',a') - \alpha \log \pi(a'|s'))$$

- Adds an entropy term to the future return estimate
- Makes the critic aware of the randomness in the actor's policy
- Encourages smoother Q-functions and more robust learning
- Used to compute soft Q-targets for critic updates

KL Divergence in Deep Reinforcement Learning

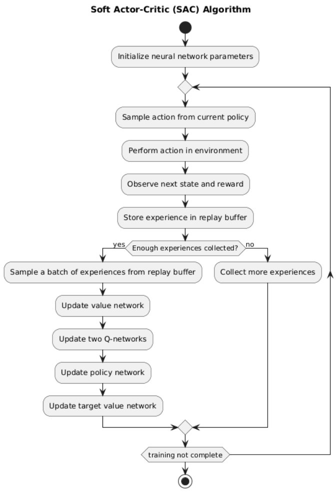
Kullback-Leibler (KL) Divergence is often used in DRL to guide or regularize policy learning.

$$\mathsf{KL}(\pi(a|s) \parallel \pi_{\mathsf{target}}(a|s)) = \mathbb{E}_\pi \left[\log \frac{\pi(a|s)}{\pi_{\mathsf{target}}(a|s)} \right]$$

- In DRL, policies are often updated by minimizing the KL divergence between the current policy and a target distribution that emphasizes better actions.
- A common choice for the target is a distribution shaped by action-values, e.g., $\pi_{\mathsf{target}}(a|s) \propto \exp(Q(s,a)/\alpha)$.
- This encourages the policy to assign higher probability to actions with higher estimated returns, while still maintaining stochasticity.
- The KL term also serves as a form of regularization, preventing abrupt or overly aggressive policy updates.

Overview of the Soft Actor-Critic Algorithm

- **Soft Actor-Critic (SAC)** is an advanced Deep Reinforcement Learning algorithm designed for continuous action spaces.
- It balances **exploration** and **exploitation** by augmenting the objective with the expected entropy — encouraging diverse actions while optimizing rewards.
- SAC extends the **Soft Policy Iteration algorithm** from tabular to continuous action spaces using parameterized state value functions modeled as expressive **neural networks**.
- SAC alternates between two main steps:
  - **Environment Interaction step**.
  - **Network Updates and Gradient step**.



Environment Interaction step in SAC

- At each time step, the agent observes the current state of the environment and **selects an action by sampling from its current policy network**.
- The environment responds by transitioning to a new state and providing a reward that **combines task reward and an entropy term**, encouraging diverse, exploratory actions.
- This transition (state, action, reward, next state) is stored in a **replay buffer** that is used to collect diverse past experiences, enabling **off-policy** learning — the agent learns from experiences generated by previous policies, not just the current one.
- Using **off-policy** data allows SAC to make efficient use of all collected data, improve **training stability**, decouple data collection from learning updates, and better handle **complex environments** where direct on-policy learning would be costly or unstable.

Network Updates and Gradient step in SAC

- After collecting experience, SAC updates its neural networks using **batches sampled from the replay buffer**.
- Three networks are trained simultaneously:
  - **Q-function networks** (two copies): Estimate the expected return for state-action pairs, minimizing the **soft Bellman residual** to reduce prediction error.
  - **Value function network**: Approximates the soft state value, trained to match the expected Q-value minus the policy's log-probability (entropy).
  - **Policy network**: Updated to minimize the **Kullback-Leibler divergence** between the current policy and an ideal distribution defined by the Q-function, encouraging actions with high expected return and entropy.
- The gradient updates use **stochastic gradient descent** and backpropagation with unbiased gradient estimators for efficient and stable training.
- A **target value network** (a slowly updated copy of the value network) is used to stabilize Q-function updates.
- This process alternates continuously with environment interaction, gradually improving policy performance.

Experimental Evaluation of Soft Actor-Critic

Experiments were conducted with the Soft Actor-Critic (SAC) algorithm to evaluate its performance in terms of sample efficiency, training stability, and overall effectiveness, especially compared to other state-of-the-art reinforcement learning algorithms.

SAC was tested on a range of continuous control tasks from:

- The OpenAI Gym suite (e.g., HalfCheetah, Hopper)
- The rllab Humanoid task, which involves controlling a 21-dimensional humanoid robot — a particularly challenging task for off-policy methods.

SAC was compared to several strong baseline algorithms:

- DDPG – a classic off-policy algorithm known for high efficiency but poor stability.
- TD3 – an enhanced version of DDPG that uses double Q-learning and delayed updates.
- PPO – a stable and widely-used on-policy policy gradient method.
- SQL – Soft Q-Learning, another entropy-regularized off-policy approach.
- Trust-PCL – a trust-region based policy consistency method.

Comparative Results of Soft Actor-Critic

SAC outperforms baseline methods, especially on more complex tasks:

- On hard benchmarks like Ant and Humanoid, SAC achieves significantly higher returns
- On easier tasks, it performs on par with other algorithms

SAC learns faster and reaches better final performance:

- Compared to DDPG, which often fails to learn at all.
- Trains more efficiently than PPO, which requires large batch sizes
- Surpasses SQL, which is slower and less effective in the long run

Overall, SAC demonstrates state-of-the-art sample efficiency, stability, and robustness across challenging benchmarks.

Metrics on the Importance of SAC Components

- **Stochastic vs. Deterministic Policy:**
  - Replacing the stochastic actor with a deterministic one (like DDPG) led to unstable training(Performance varied a lot depending on initialization and randomness for the deterministic actor).
  - The stochastic policy produced significantly more consistent results, especially on hard tasks (e.g., Humanoid).
  - Conclusion: Entropy maximization is critical for stable exploration.
- **Reward Scale Sensitivity:**
  - Reward scaling acts like inverse temperature: controls policy entropy indirectly.
  - Too small $\rightarrow$ nearly uniform policy (no learning).
    Too large $\rightarrow$ overly deterministic early on.
  - Moderate scaling yielded best results.
  - Conclusion: Reward scale is the most important/demanding hyperparameter to tune in SAC.

Metrics on the Imporance of SAC Components

Additional experiments focused on target network smoothing and evaluation strategy.

- **Target Network Update Coefficient (τ):**
  - Large τ (fast updates) $\rightarrow$ instability.
  - Small τ (slow updates) $\rightarrow$ more stable but slower learning.
  - $\tau = 0.005$ found to work reliably across environments.
  - Confirms importance of soft (Polyak) target updates.
- **Evaluation: Sampled vs. Mean Actions:**
  - Training benefits from stochastic sampling (for exploration).
  - At test time, evaluating with *mean actions* produced better returns.
  - Conclusion: Entropy helps learning, but deterministic policies exploit it best during deployment.

Conclusions

- The paper introduces Soft Actor-Critic (SAC), an off-policy deep RL algorithm that combines sample efficiency with the benefits of entropy maximization for more stable learning.
- The theoretical foundation shows that soft policy iteration converges to the optimal policy, providing a solid basis for SAC.
- Empirically, SAC outperforms state-of-the-art methods, including:
  - Off-policy DDPG — with much better sample efficiency
  - On-policy PPO — with faster and more stable learning
- The success of SAC highlights the promise of stochastic, maximum entropy RL algorithms for improved robustness and stability in continuous control.
- Future directions include exploring:
  - More advanced maximum entropy methods (e.g., trust regions)
  - More expressive policy representations