



MAY 20, 2025

INVESTIGATING ONLINE VS OFFLINE REINFORCEMENT LEARNING USING SOFT ACTOR-CRITIC (SAC)

REPORT - 01

VISWAK RB - 124104338
M.SC DATA SCIENCE & ANALYTICS
University College Cork (UCC)




Table of Contents

1.	INTRODUCTION	2
2.	EXPERIMENTAL SETUP	2
3.	RESULTS AND ANALYSIS.....	4
4.	INTERPRETATION AND TAKEAWAYS.....	6
5.	CONCLUSION	7
6.	REFERENCES.....	7

1. Introduction

This project investigates the sample efficiency and offline learning capabilities of the Soft Actor-Critic (SAC) algorithm. SAC is evaluated in two continuous control environments: **LunarLanderContinuous-v2** and **Pendulum-v1**. The goals include comparing SAC's performance across online and offline settings, understanding its learning dynamics, and identifying challenges when learning from logged data.

1.1. Soft Actor-Critic Overview

Soft Actor-Critic is an off-policy, entropy-regularized reinforcement learning algorithm designed for continuous control. It stands out due to:

- **Stochastic actor:** Outputs mean and standard deviation for actions.
- **Entropy bonus:** Encourages exploration during training.
- **Twin Q-networks:** Reduces value overestimation bias.
- **Target soft updates:** Provides stable learning updates over time.

SAC's design allows for sample-efficient learning and smooth policy updates, which makes it suitable for both online and offline RL scenarios.

2. Experimental Setup

We tested SAC on:

- **Online Learning:** Agent learns via direct environment interaction.
- **Offline Learning:** Agent learns from a fixed dataset collected during online runs.

Both setups were applied to LunarLanderContinuous-v2 and Pendulum-v1.

2.1. Environment Summary

Environment	State Dim	Action Dim	Reward Range
LunarLanderContinuous-v2	8	2	$[-\infty, \infty]$
Pendulum-v1	3	1	$[-16, 0]$

2.2. Dataset Creation

- During online runs, we saved all transitions as (state, action, reward, next_state, done).
- These datasets were later used to train the offline SAC agents.
- Saved as:
 - **lunarlander_online_dataset.pkl**
 - **pendulum_online_dataset.pkl**

2.3. Hyperparameters Used

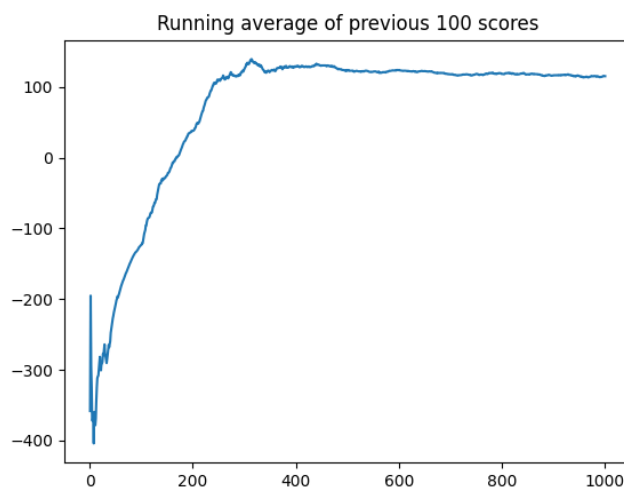
Hyperparameter	Value Used	Reason
alpha, beta	0.0003	Explicitly set for stable actor-critic learning
tau	0.005	Kept default for smooth target updates
reward_scale	2 (LunarLander), 1 (Pendulum)	Lowered for Pendulum to prevent over-amplified signals
batch_size	256	Standard for stable SAC training
hidden layers	[256, 256]	Sufficient for both environments
n_games (online)	1000	Increased to allow convergence
eval_interval (offline)	5000 (LL), 10000 (Pendulum)	For regular monitoring without overhead
n_train_steps (offline)	50k–200k	Chosen to ensure convergence on static data

3. Results and Analysis

3.1. LunarLanderContinuous-v2

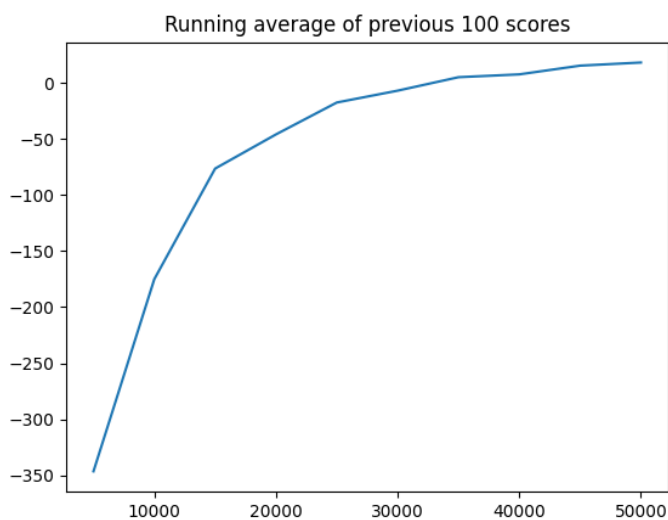
Online SAC

- After 1000 episodes, average reward reached **~+115**
- Convergence stable after 800+ episodes



Offline SAC

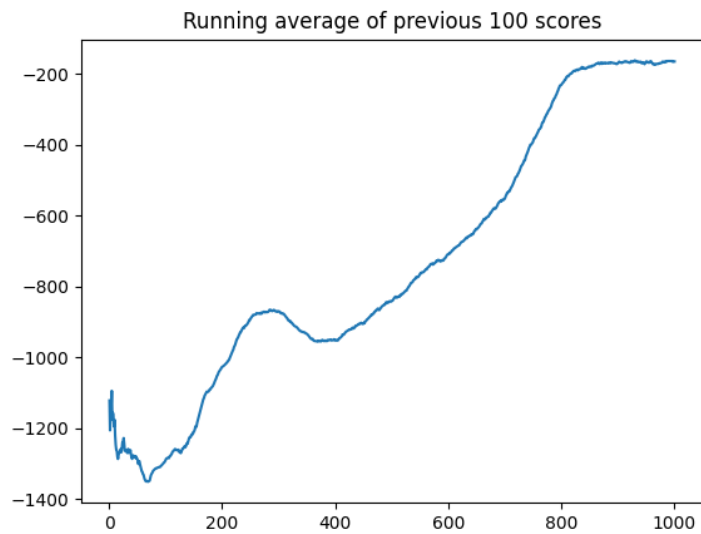
- Started at **~-346** and improved to **+43** by 50k steps
- Significant performance drop from online training



3.2. Pendulum-v1

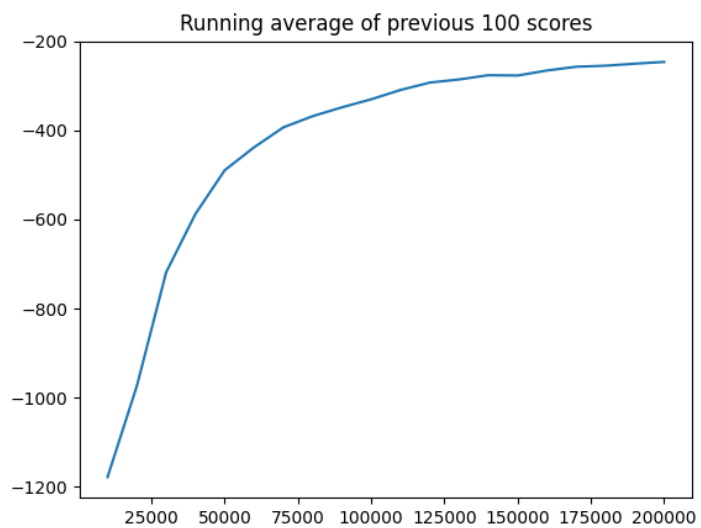
Online SAC

- Achieved reward of **-165**, which is near-optimal



Offline SAC

- Started at **-1200**, improved to **-250**
- Gap of ~85 points from online performance



4. Interpretation and Takeaways

4.1. Observations

LunarLanderContinuous-v2

- **Online** SAC performed as expected, achieving an average score above **+115**, showing stable convergence after ~800 episodes.
- **Offline** SAC trained on the collected dataset reached **~+40** average reward — **less than half** of online performance.
- The gap indicates **offline SAC struggles with unseen states**, especially in environments with more complex dynamics and long horizons.

Pendulum-v1

- **Online** SAC achieved **-165** average reward, which is close to optimal for this task.
- **Offline** SAC improved gradually from **-1200** to **-250**, showing good learning from static data.
- The offline gap (**~85 points**) was **smaller** than in LunarLander, which suggests **Pendulum's simpler dynamics** make it more forgiving.

4.2. Takeaways

Offline SAC can learn, but its performance is highly dependent on:

- **Dataset coverage:** Poor coverage leads to high generalization error
- **Reward distribution:** Environments like Pendulum (dense reward) are more offline-friendly

Stability tweaks— especially in Pendulum, where early training instability caused NaNs. We learned that:

- **Clamping $\log(\sigma)$** , adding **gradient clipping**, and applying a **random warm-up phase** helps.

Model saving and buffer population must be managed carefully, especially with large datasets and repeated evaluations.

5. Conclusion

This project showed that SAC works well in both online and offline settings, but with clear differences. Online training gave strong, stable results, while offline learning performed reasonably but couldn't match online due to the lack of exploration and fresh data. With the right data and tweaks, SAC can still learn effectively offline, though more advanced methods would be needed to fully close the gap. Looking ahead, a potential hybrid approach — combining offline pretraining with online fine-tuning — could offer the best of both.

6. References

- Offline RL with D4RL (Dataset for Datasets for Deep Data-Driven RL)

[\[https://github.com/rail-berkeley/d4rl\]](https://github.com/rail-berkeley/d4rl)(<https://github.com/rail-berkeley/d4rl>)

- Hugging Face Offline RL Tutorial

[\[https://huggingface.co/blog/offline-rl\]](https://huggingface.co/blog/offline-rl)(<https://huggingface.co/blog/offline-rl>)

- CleanRL's SAC Implementation

[\[https://github.com/vwxyzjn/cleanrl/blob/master/cleanrl/sac_continuous_action.py\]](https://github.com/vwxyzjn/cleanrl/blob/master/cleanrl/sac_continuous_action.py)

- Spinning Up by OpenAI (SAC Theory & Implementation Guide)

[\[https://spinningup.openai.com/en/latest/algorithms/sac.html\]](https://spinningup.openai.com/en/latest/algorithms/sac.html)