Soft Actor-Critic (SAC) Tutorial

September 21, 2025

1 Introduction

Soft Actor-Critic (SAC) maximizes expected return while encouraging high-entropy policies. By combining stochastic policies with Q-function learning and automatic entropy tuning, SAC achieves state-of-the-art performance on continuous control benchmarks.

2 Theory and Formulas

2.1 Maximum Entropy Objective

SAC optimizes

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\Big[\sum_t \gamma^t (r_{t+1} + \alpha\mathcal{H}(\pi(\cdot \mid s_t)))\Big], \tag{1}$$

where α balances reward and entropy.

2.2 Critic and Policy Updates

Twin Q-functions Q_{w_i} minimize

$$L(w_i) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\Big[(Q_{w_i}(s,a) - y)^2\Big], \quad y = r + \gamma(\min_j Q_{w_j^-}(s',a') - \alpha \log \pi_\theta(a' \mid s')). \tag{2}$$

The policy is updated using the reparameterization trick:

$$\nabla_\theta J_\pi = \mathbb{E}_{s\sim\mathcal{D},\varepsilon}\big[\nabla_\theta \alpha \log \pi_\theta(f_\theta(\varepsilon; s) \mid s) - \nabla_\theta Q_{w_1}(s, f_\theta(\varepsilon; s))\big]. \tag{3}$$

2.3 Temperature Adaptation

Entropy temperature α is tuned by minimizing

$$J(\alpha) = \mathbb{E}_{a\sim\pi_\theta}\big[-\alpha(\log \pi_\theta(a \mid s) + \mathcal{H}_{\text{target}})\big]. \tag{4}$$

This keeps policy entropy near a target value, automating exploration-exploitation trade-offs.

3 Applications and Tips

- **Continuous control**: locomotion, manipulation, autonomous driving.
- **Sample efficiency**: off-policy updates with replay buffers accelerate learning.
- **Robust policies**: entropy objective encourages diverse actions.
- **Best practices**: normalize observations, clip gradients, maintain twin critics, logarithmic parameterization for α, and monitor entropy as training progresses.

4 Python Practice

The script `gen_sac_figures.py` implements a lightweight SAC agent in a one-dimensional control environment. It records episode returns and the learned temperature across updates.

Listing 1: Excerpt from $\text{gen}_s ac_f igures.py$

```
q_target = reward + gamma * (min_q_target - alpha * log_prob_next)
critic_w1 += critic_lr * (q_target - q1) * features(state, action)
critic_w2 += critic_lr * (q_target - q2) * features(state, action)

alpha_grad = -np.mean(log_prob + target_entropy)
log_alpha += alpha_lr * alpha_grad
alpha = np.exp(log_alpha)
```

5 Result

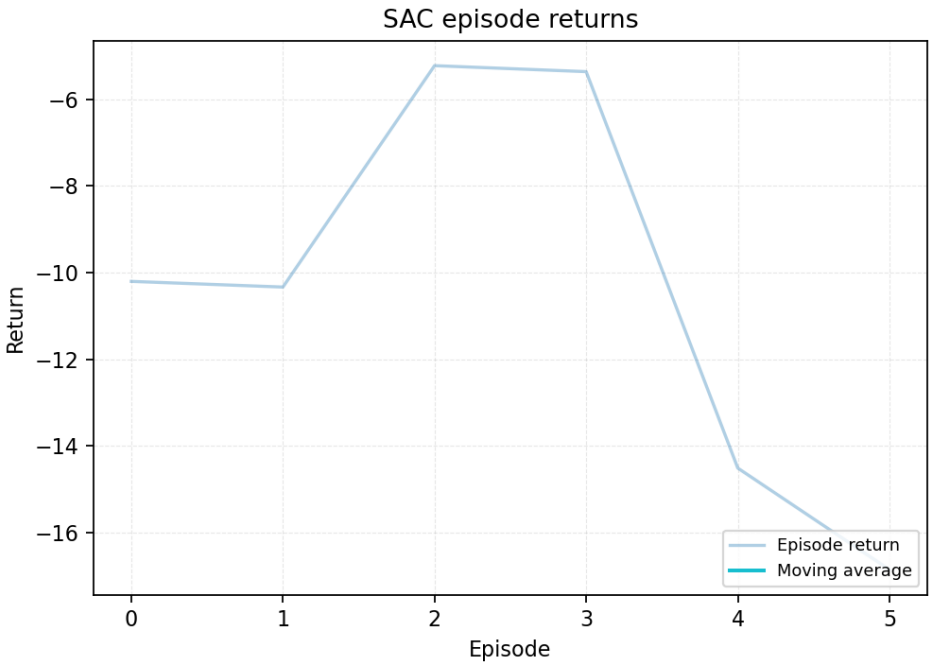

Figure 1: SAC episode returns demonstrating stable improvement

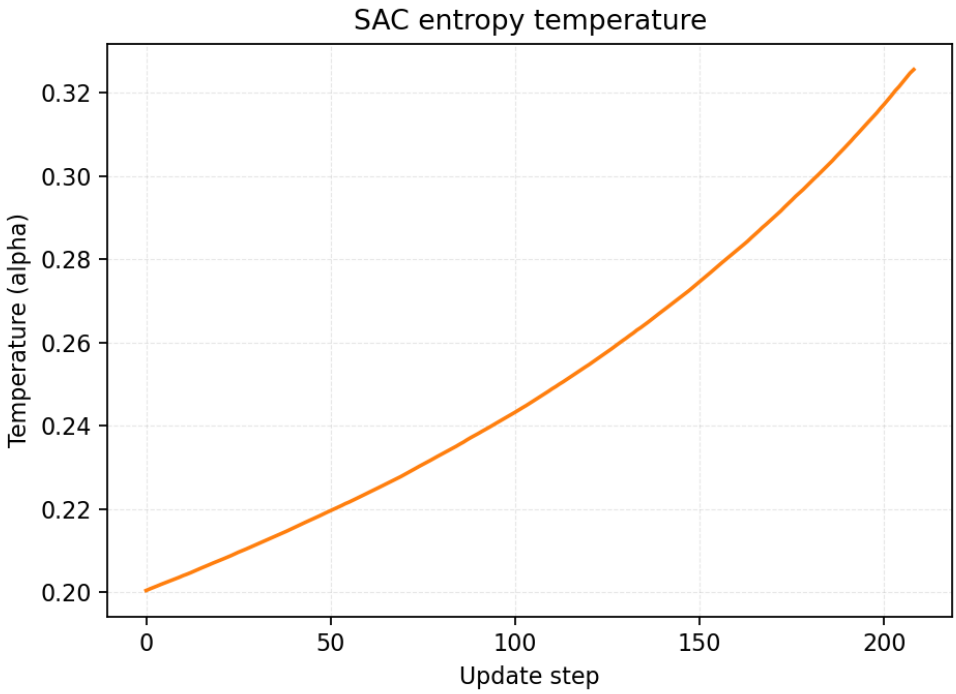

Figure 2: Entropy temperature trajectory approaching the target entropy

6 Summary

SAC maximizes reward and entropy simultaneously using twin Q-functions, stochastic actors, and automatic temperature tuning. Proper normalization, replay management, and entropy monitoring are essential. The toy control example illustrates increasing returns while the entropy temperature converges toward its target.