Soft Actor-Critic to solve the Lunar Lander Continuous Problem

Ziad Al-Ziadi and **Behzad Javaheri**

Introduction

Reinforcement learning has been successfully applied to solve a variety of problems including robotic control using different approaches [1-4]. One of the well-known robotic control problems is the lunar lander attempting to find optimal algorithms for safe planetary landing. This is interesting due to historical and recent interest in planetary exploration and colonisation. These missions require advanced navigation for accurate landing. The underpinning control algorithms need to be equipped with a real-time reconstruction of the environment obtained from onboard sensors. Historically, this has been achieved using off-line mechanisms computed beforehand to guide the controller. Recent advances in RL have allowed closed-loop algorithms to be trained in simulation beforehand to obtain the optimal capability to satisfy the increasingly stringent accuracy requirements for landing. OpenAI Gym toolkit provides a variety of RL environments including Lunar Lander [5]

In this task we will implement a recently described algorithm [6] and will test and analyse its performance on OpenAI continuous Lunar lander, to obtain optimal landing strategy. Previous studies have used a variety of algorithms for solving the Lunar lander problem [7-9].

Soft Actor-Critic (SAC)

SAC is an off-policy algorithm used for environments that incur continuous action spaces. In SAC the policy is trained to maximise the trade-off between the agent's expected return and randomness in the policy (entropy). Central to SAC is entropy regularisation allowing the agent to maximise the entropy in a policy in addition to maximising its cumulative rewards. Achieving high entropy in SAC will allow the policy to promote exploration [6].

$$V^{\pi}(s) = \mathop{\mathrm{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}\Big(R(s_t,a_t,s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right)\Big)\,\middle|\, s_0 = s\right]$$

Figure 1. Equation describing V^{π} of Soft Actor Policy

Maximum entropy is prioritised in SAC to encourage the policy to assign equal probabilities to actions and to prevent the policy to exploit inconsistencies in the Q function [10]. Haarnoja *et al*., use V^{π} as the value function to include *entropy bonuses* for every timestep as shown in Figure 1 where H denotes the entropy. Furthermore, Q^{π} is also adjusted to include the same entropy bonuses [6] (Fig. 2).

$$Q^{\pi}(s,a) = \mathop{\mathrm{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}R(s_t,a_t,s_{t+1}) + \alpha\sum_{t=1}^{\infty}\gamma^{t}H\left(\pi(\cdot|s_t)\right)\,\middle|\, s_0 = s, a_0 = a\right]$$

Figure 2. Description of Q-function for Soft Actor Critic

The SAC's value and Q-functions are different compared to conventional off-policy algorithms primarily due to entropy regularisation (Figs. 1-2). The SAC's objective function incorporates H ultimately to maximise the trajectory return. Moreover, the objective function maximises both reward and entropy terms allowing policy, and ultimately the agent, to explore new actions whilst maintaining a maximum reward (Fig. 3) [6, 10].

$$\pi^{*} = \arg\max_{\pi} \mathop{\mathrm{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}\Big(R(s_t,a_t,s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right)\Big)\right]$$

Figure 3. Description of objective function for SAC

In addition, because SAC is a model-free algorithm it needs to store experiences (transitions) as samples for the agent to use later. The replay buffer acts like memory and stores states, actions, rewards and next states which the agent samples from in a form of a given batch size [6].

Description of the environment

The environment used for this task is Lunar Lander and the framework is OpenAI gym [5]. The simulator employed is Box2D and the environment ID is LunarLanderContinuous-v2 (Fig. 4). Similar to real physics state space is continuous, however, action space is discrete. The main objective is to land a spacecraft between two static flag poles on the moon surface without crashing and softly and fuel-efficiently.

The observation space determines the lander's attributes. Eight state variables are associated with the state space, 1) x coordinate and; 2) y coordinate of the lander; 3) the horizontal (v_x) and; 4) the vertical velocity (v_y); 5) the orientation in space (θ); 6) the angular velocity (v_θ); 7) right leg and 8) left leg touching the ground [5].

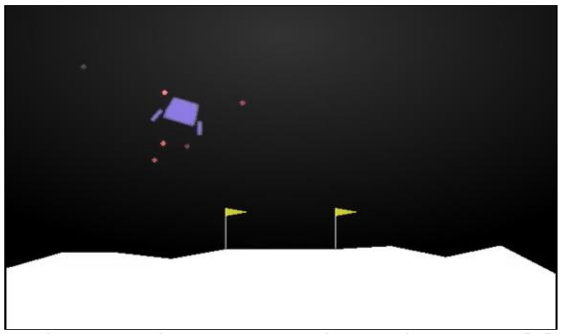

Figure 4. The Lunar Lander environment. [5]

All the coordinate values are provided relative to the landing pad instead. The landing pad is always at coordinates (0,0). The lander has 4 possible discrete actions: to fire left, right, bottom engine or do nothing. The fuel is infinite and to obtain rewards the agent needs to maintain both a good posture mid-air and reach the landing pad as quickly and accurately as possible. For the Lunar Lander environment, the reward is defined as:

$$Reward(st) = -100 * (dt - dt-1) - 100 * (vt - vt-1) - 100 * (\omega t - \omega t-1) + hasLanded(st)$$ [5, 11].

where dt is the distance to the landing pad, v_t and ωt are the agent's velocity angular velocity at time t, respectively. The reward function is denoted by hasLanded() which describes boolean state values that represent the information on whether the lander lost contact with the landing pad and if landed softly. For navigating to the landing area, the agent is awarded a reward between 100-140. Moving away from the landing pad will lead to receiving a negative reward. Two terminal states of successful landing and crash are defined with rewards of +100 or -100 points, respectively. Each leg ground contact is +10, firing the main engine is -0.3 points each frame and action is two real values vectors from -1 to +1 [5].

Case study

We have devised the implementation of SAC into 6 sections:

1. *Defining 3 networks [actor (policy), critic and value]*: initially, the number of actions, hidden layers and initial weight are defined. We allow using "Cuda" to speed up computation. Two outputs of mean and the log standard deviation have been defined for the actor with the "Relu" activation function. The log standard deviation is clamped, and we employ reparameterisation to obtain the actions. To achieve this, noise from a standard normal distribution is sampled and multiplied with standard deviation and the result is added to the mean. This value is activated with a "Tanh" function to provide action. An Approximator of the log-likelihood of Tanh(mean + std* z) was used to compute the log probability.

2. *Construction of agent's memory (replay buffer)*: information about the dimension of the observations of the environment and the action space to be stored in memory.

3. *Setting hyperparameters*: hyperparameters of our networks are defined [6].

4. *Initialising the networks*: three networks defined in step 1 and a target V network initialised.

5. *Initialising environment*: OpenAI Gym LunarLanderContinuous-v2 environment is initialised.

6. *Training and evaluation*: our model starts training and a dynamic graph corresponding to reward for every n episode plotted in a nested loop. The beginning of an episode is initialised by the outer loop. The inner loop is to implement individual steps within an episode. Within this inner loop action from the actor-network, or randomly from the action space for the first few time steps, is sampled and the state, action, reward, next state, and terminal are recorded.

This loop stops when a minimum number of observations in the buffer is recorded.

Performance evaluation

After running our SAC on 120 episodes, our model achieved the highest reward of 78.53. We observed that rewards the agent obtained fluctuated heavily in that the agent first obtains negative rewards then slowly increases total rewards before reverting into negative rewards to follow the same pattern (Fig. 5A). Parameter values are illustrated in Table 1.

Parameter	Value
$\gamma_{Discount\ Factor}$	0.99
$\alpha_{Learning\ Rate}$	0.0003
τ_{Tau}	0.005

Table 1: Initial SAC parameter values

Performance with different parameters

We aimed to evaluate the model using significantly different parameter values (Table 2) to examine the effect on the performance. Our data show that reducing discount factor, learning rate and tau have negatively impacted the performance with significant fluctuations between episodes, a trend evident throughout the training process. With the original parameters (Fig. 5A) we observe a trend in which the performance is enhanced with continued training, this is, however, lost in the model with modified parameters (Fig. 5B). It appears that our modified model does not learn from experience impacted by reduced future rewards resulting from the lower discount factor and learning rate which does not allow for Q-values to be updated as needed. In the modified model, the agent follows virtually no pattern and instead, seems to engage in random learning and fluctuating rewards.

Parameter	Value
$\gamma_{Discount\ Factor}$	0.33
$\alpha_{Learning\ Rate}$	0.05
τ_{Tau}	0.5

Table 2: Modified SAC parameter values

Previous studies aiming to solve the Lunar Lander problem have utilised a variety of approaches. For example, Banerjee et al (2019) used a modified policy gradient approach [12]. In addition, Lu et al., (2019) utilised a control-model-based method which instead of the dynamics of the system it learns the optimal control parameters [13]. Whilst in the original SAC implementation various environments including Hopper, Walker2d, HalfCheetah, Ant and Humanoid were tested, no implementation on the Lunar Lander was reported [6].

Our findings are consistent with a recent report implementing SAC and other algorithms to solve a series of problems including the Lunar lander Continuous environment. We believe

that the performance can further be enhanced by an exhaustive grid search based hyperparameter tuning.

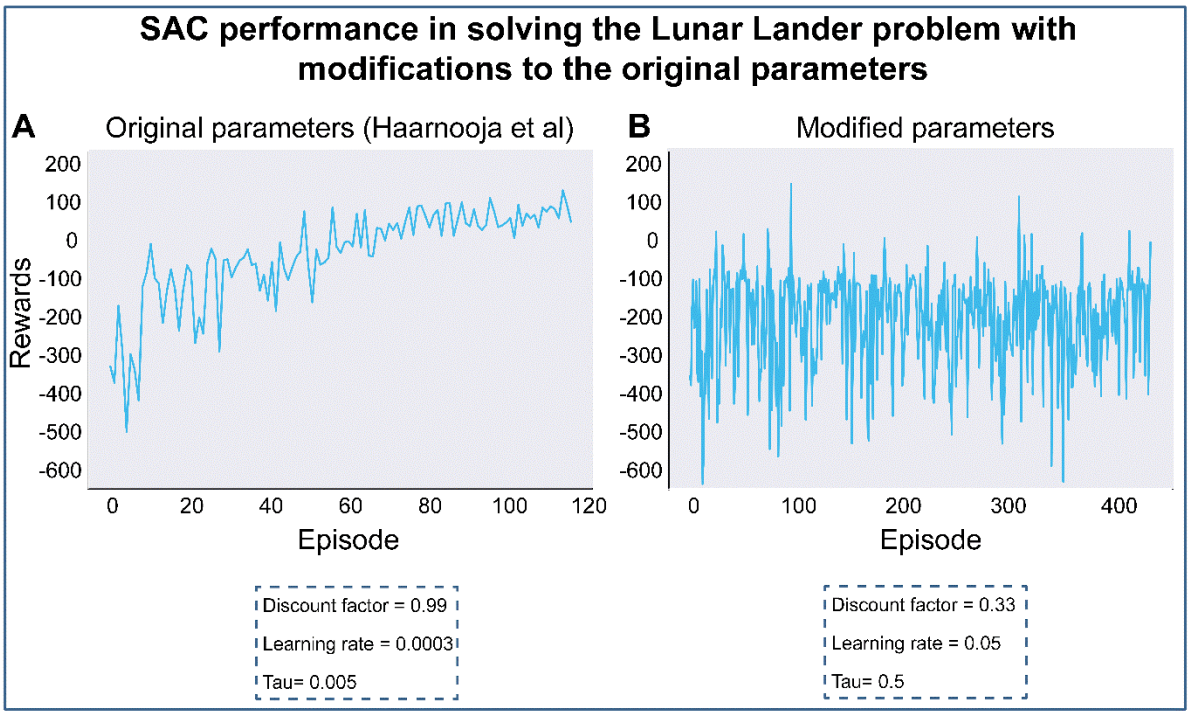

Figure 5. The performance of SAC for solving the Lunar lander problem with the original and modified parameters. A) SAC performance with parameters suggested by Haarnoja *et al*., and B) the effect of a modified discount factor, learning rate and Tau on SAC performance.

References

[1] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research,* vol. 32, no. 11, pp. 1238-1274, 2013.

[2] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange, "Reinforcement learning for robot soccer," *Autonomous Robots,* vol. 27, no. 1, pp. 55-73, 2009.

[3] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis lectures on artificial intelligence and machine learning,* vol. 4, no. 1, pp. 1-103, 2010.

[4] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research,* vol. 4, pp. 237-285, 1996.

[5] G. Brockman *et al.*, "Openai gym," *arXiv preprint arXiv:1606.01540,* 2016.

[6] T. Haarnoja *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905,* 2018.

[7] X.-L. Liu, G.-R. Duan, and K.-L. Teo, "Optimal soft landing control for moon lander," *Automatica,* vol. 44, no. 4, pp. 1097-1103, 2008.

[8] D.-H. Cho, B.-Y. Jeong, D.-H. Lee, and H.-C. Bang, "Optimal perilune altitude of lunar landing trajectory," *International Journal of Aeronautical and Space Sciences,* vol. 10, no. 1, pp. 67-74, 2009.

[9] T. Brady and S. Paschall, "The challenge of safe lunar landing," in *2010 IEEE Aerospace Conference*, 2010: IEEE, pp. 1-14.

[10] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*, 2018: PMLR, pp. 1861-1870.

[11] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[12] A. Banerjee, D. Ghosh, and S. Das, "Evolving Network Topology in Policy Gradient Reinforcement Learning Algorithms," in *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, 2019: IEEE, pp. 1-5.
[13] Y. Lu, M. S. Squillante, and C. W. Wu, "A Control-Model-Based Approach for Reinforcement Learning," *arXiv preprint arXiv:1905.12009,* 2019.