Soft Actor Critic—Deep Reinforcement Learning with Real-World Robots

Tuomas Haarnoja, *Vitchyr Pong*, *Kristian Hartikainen*,
Aurick Zhou, *Murtaza Dalal*, *and* *Sergey Levine*
Dec 14, 2018

We are announcing the release of our state-of-the-art off-policy model-free reinforcement learning algorithm, soft actor-critic (SAC). This algorithm has been developed jointly at UC Berkeley and Google, and we have been using it internally for our robotics experiment. Soft actor-critic is, to our knowledge, one of the most efficient model-free algorithms available today, making it especially well-suited for real-world robotic learning. In this post, we will benchmark SAC against state-of-the-art model-free RL algorithms and showcase a spectrum of real-world robot examples, ranging from manipulation to locomotion. We also release our implementation of SAC, which is particularly designed for real-world robotic systems.

Desired Features for Deep RL for Real Robots

What makes an ideal deep RL algorithm for real-world systems? Real-world experimentation brings additional challenges, such as constant interruptions in the data stream, requirement for a low-latency inference and smooth exploration to avoid mechanical wear and tear on the robot, which set additional requirement for both the algorithm and also the implementation of the algorithm.

Regarding the algorithm, several properties are desirable:

- **Sample Efficiency**. Learning skills in the real world can take a substantial amount of time. Prototyping a new task takes several trials, and the total time required to learn a new skill quickly adds up. Thus good sample complexity is the first prerequisite for successful skill acquisition.
- **No Sensitive Hyperparameters**. In the real world, we want to avoid parameter tuning for the obvious reason. Maximum entropy RL provides a robust framework that minimizes the need for hyperparameter tuning.
- **Off-Policy Learning**. An algorithm is off-policy if we can reuse data collected for another task. In a typical scenario, we need to adjust parameters and shape the reward function when prototyping a new task, and use of an off-policy algorithm allows reusing the already collected data.

Soft actor-critic (SAC), described below, is an off-policy model-free deep RL algorithm that is well aligned with these requirements. In particular, we show that it is sample efficient enough to solve real-world robot tasks in only a handful of hours, robust to hyperparameters and works on a variety of simulated environments with a single set of hyperparameters.

In addition to the desired algorithmic properties, experimentation in the real-world sets additional requirements for the implementation. Our release supports many of these features that we have found crucial when learning with real robots, perhaps the most importantly:

- **Asynchronous Sampling**. Inference needs to be fast to minimize delay in the control loop, and we typically want to keep training during the environment resets too. Therefore, data sampling and training should run in independent threads or processes.
- **Stop / Resume Training**. When working with real hardware, whatever can go wrong, will go wrong. We should expect constant interruptions in the data stream.
- **Action smoothing**. Typical Gaussian exploration makes the actuators jitter at high frequency, potentially damaging the hardware. Thus temporally correlating the exploration is important.

Soft Actor-Critic

Soft actor-critic is based on the maximum entropy reinforcement learning framework, which considers the entropy augmented objective

$$J(\pi) = \mathbb{E}_\pi \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log(\pi(\mathbf{a}_t|\mathbf{s}_t)) \right],$$

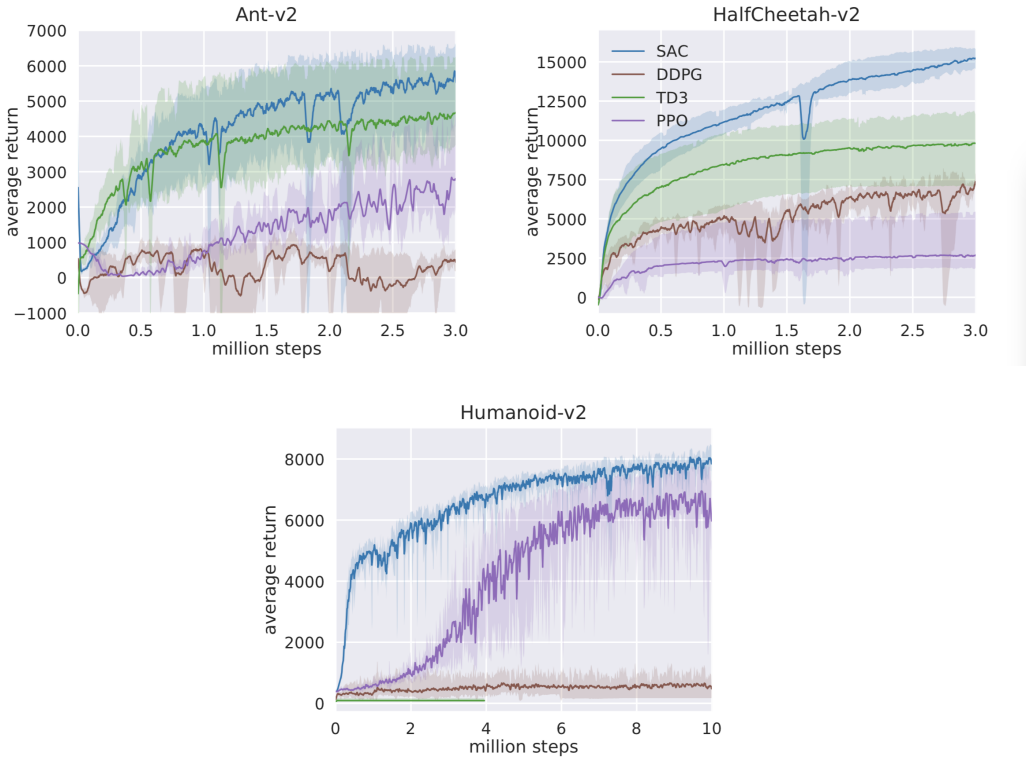
where $\mathbf{s}_t$ and $\mathbf{a}_t$ are the state and the action, and the expectation is taken over the policy and the true dynamics of the system. In other words, the optimal policy not only maximizes the expected return (first summand) but also the expected entropy of itself (second summand). The trade-off between the two is controlled by the non-negative temperature parameter α, and we can always recover the conventional, maximum expected return objective by setting $\alpha = 0$. In a technical report, we show that we can view this objective as an entropy constrained maximization of the expected return, and learn the temperature parameter automatically instead of treating it as a hyperparameter.

This objective can be interpreted in several ways. We can view the entropy term as an uninformative (uniform) prior over the policy, but we can also view it as a regularizer or as an attempt to trade off between exploration (maximize entropy) and exploitation (maximize return). In our previous post, we gave a broader overview and proposed applications that are unique to maximum entropy RL, and a probabilistic view of the objective is discussed in a recent tutorial. Soft actor-critic maximizes this objective by parameterizing a Gaussian policy and a

Q-function with a neural network, and optimizing them using approximate dynamic programming. We defer further details of soft actor-critic to the technical report. In this post, we will view the objective as a grounded way to derive better reinforcement learning algorithms that perform consistently and are sample efficient enough to be applicable to real-world robotic applications, and—perhaps surprisingly—can yield state-of-the-art performance under the conventional, maximum expected return objective (without entropy regularization) in simulated benchmarks.

Simulated Benchmarks

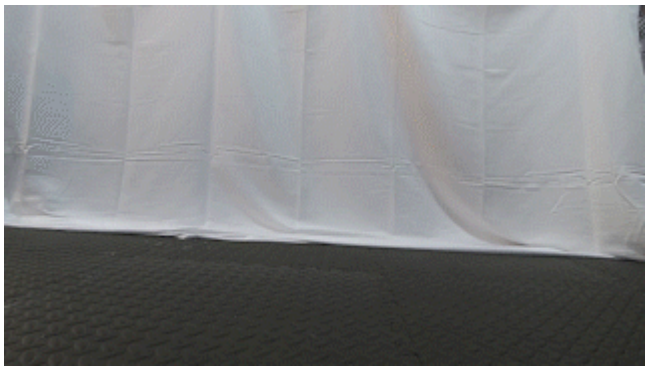
Before we jump into real-world experiments, we compare SAC on standard benchmark tasks to other popular deep RL algorithms, deep deterministic policy gradient (DDPG), twin delayed deep deterministic policy gradient (TD3), and proximal policy optimization (PPO). The figures below compare the algorithms on three challenging locomotion tasks, HalfCheetah, Ant, and Humanoid, from OpenAI Gym. The solid lines depict the total average return and the shadings correspond to the best and the worst trial over five random seeds. Indeed, soft actor-critic, which is shown in blue, achieves the best performance, and—what's even more important for real-world applications—it performs well also in the worst case. We have included more benchmark results in the technical report.



Deep RL in the Real World

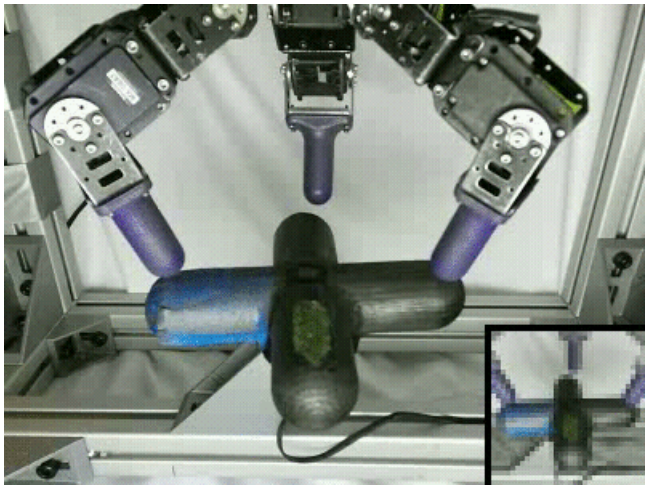
We tested soft actor-critic in the real world by solving three tasks from scratch without relying on simulation or demonstrations. Our first real-world task involves the Minitaur robot, a small-scale quadruped with eight direct-drive actuators. The action space consists of the swing angle and the extension of each leg, which are then mapped to desired motor positions and tracked with a PD controller. The observations include the motor angles as well as roll and pitch angles and angular velocities of the base. This learning task presents substantial challenges for real-world reinforcement learning. The robot is underactuated, and must therefore delicately

balance contact forces on the legs to make forward progress. An untrained policy can lose balance and fall, and too many falls will eventually damage the robot, making sample-efficient learning essentially. The video below illustrates the learned skill. Although we trained our policy only on flat terrain, we then tested it on varied terrains and obstacles. Because soft actor-critic learns robust policies, due to entropy maximization at training time, the policy can readily generalize to these perturbations without any additional learning.



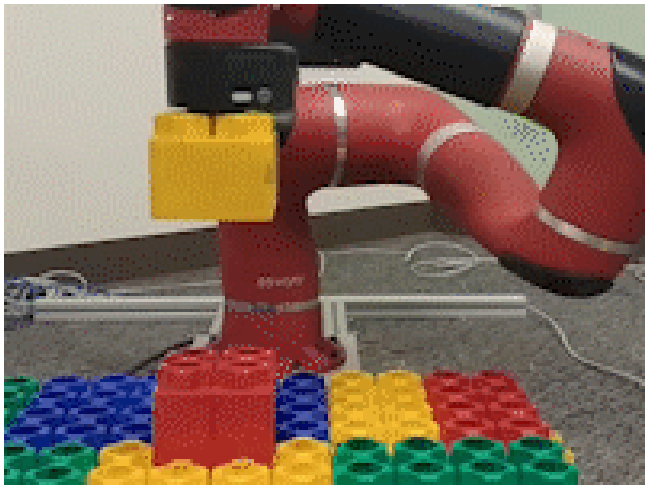
The Minitaur robot (Google, Tuomas Haarnoja, Sehoon Ha, Jie Tan, and Sergey Levine).

Our second real-world robotic task involves training a 3-finger dexterous robotic hand to manipulate an object. The hand is based on the Dynamixel Claw hand, discussed in another post. This hand has 9 DoFs, each controlled by a Dynamixel servo-motor. The policy controls the hand by sending target joint angle positions for the on-board PID controller. The manipulation task requires the hand to rotate a ``valve''-like object as shown in the animation below. In order to perceive the valve, the robot must use raw RGB images shown in the inset at the bottom right. The robot must rotate the valve so that the colored peg faces the right (see video below). The initial position of the valve is reset uniformly at random for each episode, forcing the policy to learn to use the raw RGB images to perceive the current valve orientation. A small motor is attached to the valve to automate resets and to provide the ground truth position for the determination of the reward function. The position of this motor is not provided to the policy. This task is exceptionally challenging due to both the perception challenges and the need to control a hand with 9 degrees of freedom.



Rotating a valve with a dexterous hand, learned directly from raw pixels (UC Berkeley, Kristian Hartikainen, Vikash Kumar, Henry Zhu, Abhishek Gupta, Tuomas Haarnoja, and Sergey Levine).

In the final task, we trained a 7-DoF Sawyer robot to stack Lego blocks. The policy receives the joint positions and velocities, as well as end-effector force as an input and outputs torque commands to each of the seven joints. The biggest challenge is to accurately align the studs before exerting a downward force to overcome the friction between them.



Stacking Legos with Sawyer (UC Berkeley, Aurick Zhou, Tuomas Haarnoja, and Sergey Levine).

Soft actor-critic solves all of these tasks quickly: the Minitaur locomotion and the block-stacking tasks both take 2 hours, and the valve-turning task from image observations takes 20 hours. We also learned a policy for the valve-turning task without images by providing the actual valve position as an observation to the policy. Soft actor-critic can learn this easier version of the valve task in 3 hours. For comparison, prior work has used PPO to learn the same task without images in 7.4 hours.

Conclusion

Soft actor-critic is a step towards feasible deep RL with real-world robots. Work still needs to be done to scale up these methods to more challenging tasks, but we believe we are getting closer to the critical point where deep RL can become a practical solution for robotic tasks. Meanwhile, you can connect your robot to our toolbox and get learning started!

Acknowledgements

We would like to thank the amazing teams at Google and UC Berkeley—specifically Pieter Abbeel, Abhishek Gupta, Sehoon Ha, Vikash Kumar, Sergey Levine, Jie Tan, George Tucker, Vincent Vanhoucke, Henry Zhu—who contributed to the development of the algorithm, spent long days running experiments, and provided the support and resources that made the project possible.

Links:

- Project website
- Technical description of SAC
- softlearning (our robot learning toolbox, including a SAC implementation in Tensorflow)
- rlkit (another SAC implementation from UC Berkeley in PyTorch)

Subscribe to our RSS feed.
Spread the word:

Comments

BAIR Blog Comment Policy

Please be civil and only engage in academic, professional discussions that are relevant to the blog post.

Please read our Comment Policy before commenting.

6 Comments BAIR Blog 1 Login

Recommend 4 Tweet Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Manuel Haß • 9 months ago

Very cool progress! However I'm confused by the different implementations:

In Haarnojas TF implementation, there are 2 Q-networks (like described in the paper) and only the Value-network maintains a target network.
The softlearning TF implementation is a minimal reduction to the SAC idea, in that the value is computed from the Q-network and the policy, while maintaining a Q target network.
But the rlkit (pytorch) wording is not that clear. The SAC implementation uses 1 Q-network and 1 V-network, which deviates from the paper and isn't accounting for the policy bias anymore. The "Twin SAC" implementation, described as "Combination of SAC and TD3" has 2 Q-networks and Value-network (with target), which is identical(!?) to the Haarnoja version from the paper.

I understand the softlearning version as a SAC baseline.
For the rlkit implementation I see the urge to separate the ideas, but find the wording confusing.

Wouldn't a "Combination of SAC and TD3" also maintain an actor target? or isn't this necessary since the entropy term acts as an regularizer/stabilizer anyway?

2 • Reply • Share ›

Shamane Siriwardhana • 9 months ago

Any comparison with IMPALA ?

2 • Reply • Share ›

Lucas Vazquez ➚ Shamane Siriwardhana • 9 months ago

IMPALA is a framework for distributed learning when we have access to multiple parallel environments.
We only have one robot, IMPALA is not a great choice here =/

1 ^ | v • Reply • Share ›

Abdul Rahman Dabbour • 9 months ago

I'd like to cite the technical description of SAC in a class project; however I can't find it anywhere except this blog post. Could you perhaps add an arXiv link? Thank you!

PS: Wow, awesome job!

^ | v • Reply • Share ›

Jianghao Huo • 9 months ago • edited

OMG, what a nice progress! Though append the open source page link will make the post better ;)

^ | v • Reply • Share ›

AutoSniper ➚ Jianghao Huo • 9 months ago

It's already included. Find the "softlearning" and "rlkit" links at the end of the article. It's not hard.

^ | v • Reply • Share ›

ALSO ON **BAIR BLOG**

Model-Based Reinforcement Learning from Pixels with

Assessing Generalization in Deep Reinforcement Learning