Trust Region Optimization of Optimistic Actor Critic

Niklas Kappes [1] **Pascal Herrmann** [1]

Abstract

The exploration-exploitation trade-off is a fundamental challenge in reinforcement learning. While off-policy algorithms like *Soft Actor-Critic* (SAC) yield good performance, they can struggle with data efficient exploration. *Optimistic Actor-Critic* (OAC) builds upon SAC by improving the exploration behavior. But the resulting policy often samples actions at the action boundaries, which is not desirable if the policy should be deployed to a real system.

We introduce *Trust Region Optimistic Actor-Critic* (TROAC), a novel algorithm that interpolates between the exploration behavior of SAC and OAC by using trust region optimization. The resulting policy is less likely to sample actions at the limits while reaching a similar or better performance than OAC.

1. Introduction

Reinforcement learning (RL) (Sutton & Barto, 1998) is the task of improving the policy of an agent by learning from past experience. Control tasks in robotics and similar domains can become quite complicated since these systems usually have many degrees of freedom. It is therefore very desirable to learn a policy for the system instead of tuning it by hand. This makes RL to an indispensable discipline in the era of machine learning.

On one hand, a RL agent can explore the state-action space in an attempt to find a better policy. But during exploration, the agent usually does not receive that many rewards. On the other hand, the agent can exploit its current policy and gain rewards by it. But the current policy might not be optimal and the agent performs worse in the long run. This dilemma is called the exploration-exploitation trade-off. Ideally, the agent should find the optimal policy as fast as possible and then exploit it.

While current approaches have made a lot of progress in this area, a lot of work is still left to be done. One recent algorithm that tries to balance exploration and exploitation is *Soft Actor-Critic* (SAC) (Haarnoja et al., 2018). SAC approaches the task of reinforcement learning by using a maximum entropy framework. Here, the agent should not only maximize its reward but also maximize its entropy to ensure exploration. But as (Ciosek et al., 2019) have shown, SAC suffers from pessimistic underexploration and directional uninformedness. They try to solve these problems by introducing *Optimistic Actor-Critic* (OAC). OAC uses a separate exploration policy of which the mean is shifted compared to the current policy. The length of the shift is determined by a *Kullback-Leibler* (KL) constraint.

In this paper, we show that while OAC does improve the performance of SAC, it also results in a bang-bang policy, meaning that the actions are sampled at the boundaries of the action space. This behavior is not desirable since it can damage a real system if deployed to it. Furthermore, OAC does not check if the exploration policy actually samples better actions than the current policy. This can lead to suboptimal actions and inhibit improvement of the agent. To tackle these issues, we introduce *Trust Region Optimistic Actor Critic* (TROAC). TROAC uses trust region optimization techniques to guarantee actions that are better or at least not worse than those of the current policy. This also allows TROAC to interpolate between the current policy of SAC, and the exploration policy of OAC. We show that this also reduces the likelihood of the agent to result in a bang-bang policy.

We start by giving an overview of related work in section 2. We continue by looking at the theoretical background for this paper in section 3. Here, we also analyze the tendency of OAC to result in a bang-bang policy. In section 4, we introduce TROAC and analyze its performance as well as its tendency to produce band-bang policies in section 5. Then, we discuss our results in section 6, where we give an outlook about future work as well. Finally, we give a conclusion in section 7.

[*]Equal contribution [1]Department of Computer Science, Technische Universität Darmstadt, Germany. Correspondence to: Niklas Kappes <niklas.kappes@stud.tu-darmstadt.de>, Pascal Herrmann <pascal.herrmann1@stud.tu-darmstadt.de>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

2. Related Work

2.1. Off-Policy

State of the art off-policy reinforcement learning algorithms achieve high gains by reusing past experiences stored in a replay buffer (Haarnoja et al., 2018) (Lillicrap et al., 2015). But off-policy algorithms have to deal with several problems and difficulties. Learning the Q-function leads to instability and overestimation errors due to recursively applying the Bellman backup on the learned target function (Fujimoto et al., 2018). Several different techniques are used to deal with these problems. Double Q-Learning (Hasselt, 2010) and furthermore clipped double Q-learning (Fujimoto et al., 2018) deal with the overestimation error in continuous control tasks and stabilises Q-learning. Additionally a maximum entropy objective (Haarnoja et al., 2019) enables stability and leads to policies with better exploration capabilities.

We use off-policies algorithms for our approach, because different policies can be used for exploration and exploitation (Whitney et al., 2021). By that, we can effectively explore the state-action space without risking that the exploitation policy gets stuck in a local maximum.

2.2. Exploration Strategies

The close connection between sample efficiency and exploration results in different exploration approaches to improve the overall learning process. Current algorithms use an information gain given current beliefs (Russo & Roy, 2014), posterior sampling, also known as Thompson sampling (Chapelle & Li, 2011), and optimistic exploration strategies, like the *upper confidence bound* (UCB) (Auer et al., 2002), to explore new actions while enforcing exploitation to achieve higher rewards. Furthermore decoupling the exploration and exploitation policy (Whitney et al., 2021) improves pessimistic under-exploration caused by stabilizing Q-learning (Ciosek et al., 2019) and directional uninformedness of exploration strategies by locally optimizing the Q-function (Ciosek et al., 2019) or using Measured-Valued derivatives (Carvalho et al., 2021).

3. Background

3.1. Reinforcement Learning

In *Reinforcement learning* (RL), we consider an infinite *Markov Decision Process* (MDP) (Puterman, 2014), defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where an agent observes the current environment state $s_t \in \mathcal{S}$ and performs an action $a_t \in \mathcal{A}$. This leads to a new environment state $s_{t+1} \sim p(\cdot|s_t, a_t)$, where $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$ represents the probability density of state transitions. For each transition in the environment the agent receives a reward $r : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$. The overall goal of the agent is to maximize the total expected reward $J = \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t, a_t)]$, where $\rho_\pi(s_t)$ and $\rho_\pi(s_t, a_t)$ denote state and state-action marginals of the trajectory distribution produced by the policy $\pi(a_t|s_t)$ and γ defines the discount factor. Accordingly, a trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ...)$ is obtained by sequentially generated actions from the policy (Sutton & Barto, 1998).

3.2. OAC

Optimistic Actor-Critic (OAC) (Ciosek et al., 2019), an optimistic exploration algorithm based on *Soft Actor Critc* (SAC) (Haarnoja et al., 2018), addresses the two problems of pessimistic underexploration and directional uninformedness. Both problems occur for state-of-the-art actor-critic algorithms that use clipped Q-learning to reduce overestimation bias by computing an approximate lower confidence bound (Hasselt, 2010) (Van Hasselt et al., 2016) and additionally use the same policy for exploration and exploitation, as done in *Soft Actor Critic* (SAC) (Haarnoja et al., 2018) and *Twin Delayed Deep Deterministic* (TD3) (Fujimoto et al., 2018).

In OAC, an upper confidence bound $\hat{Q}_{UB}$ is used to achieve an optimistic exploration strategy. The epistemic uncertainty is modeled by a Gaussian represented by the mean $\mu_Q(s, a)$ and variance $\sigma_Q(s, a)$ of the two Q-function approximations $Q^{\{1,2\}}$, that are learend by clipped Q-learning. The level of optimism is controlled by the hyperparameter β_{UB} and accordingly the exploration behavior.

$$\hat{Q}_{UB}(s, a) = \mu_Q(s, a) + \beta_{UB}\sigma_Q(s, a) \tag{1}$$

OAC improves the exploration/exploitation trade-off by maximizing the linear approximated upper confidence bound $\bar{Q}_{UB}(s, a)$ on the state-action value function to obtain a better exploration policy (Brafman & Tennenholtz, 2002).

$$\begin{aligned} \mu_E, \Sigma_E = \arg\max_{\mu,\Sigma} \mathbb{E}_{a\sim\mathcal{N}(\mu,\Sigma)}\left[\bar{Q}_{UB}(s, a)\right] \\ \text{s. t. } \mathcal{KL}(\mathcal{N}(\mu, \Sigma), \mathcal{N}(\mu_T, \Sigma_T)) \leq \delta \end{aligned} \tag{2}$$

The UCB is linearly approximated around the target policy π_T and additionally a *Kullback-Leibler* (KL) divergence is used to bound the optimization problem and preserve stability of the optimization and update. This leads to a closed-loop decoupled exploration policy, that is optimistic, directional informed and represented by a Gaussian policy π_E:

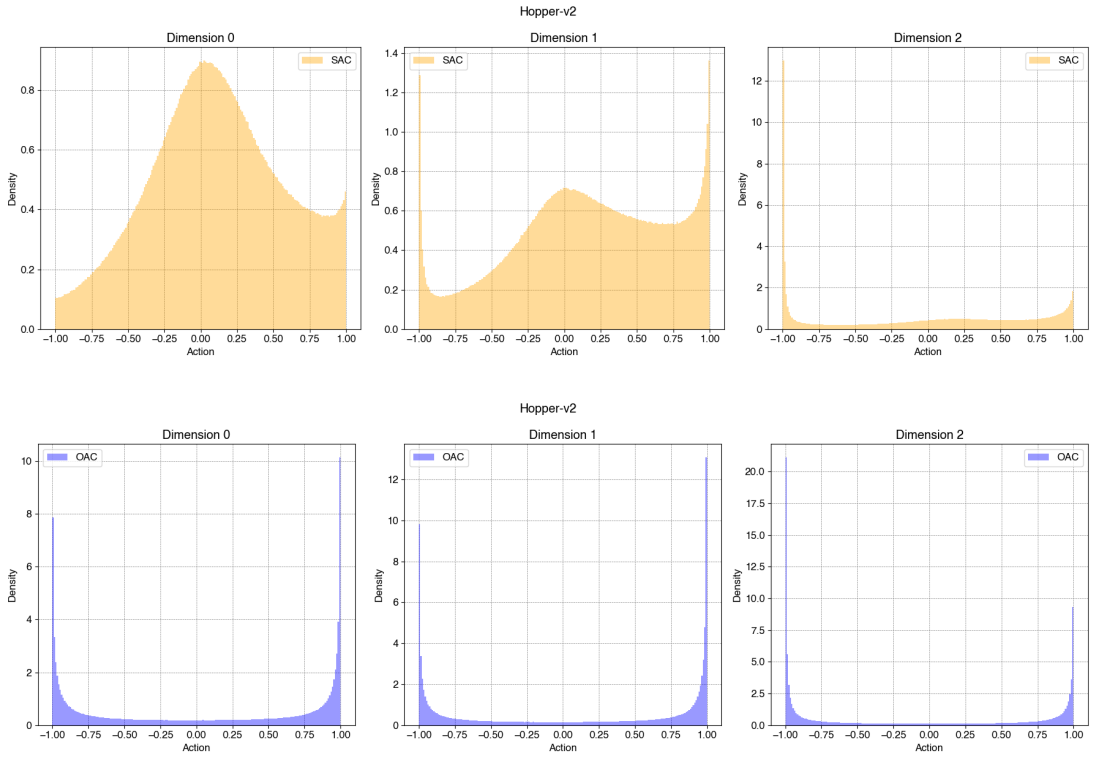

Figure 1. Histogram of sampled actions of SAC and OAC, evaluated in *Hopper-v2* environment for 5 random seeds and hyperparameters described in Appendix B. The x-axis show the action space while the y-axis corresponds to the density.

$$\mu_E = \mu_T + \sqrt{2\delta}\frac{\Sigma_T\left[\nabla_a \hat{Q}_{UB}(s,a)\right]_{a=\mu_T}}{||\left[\nabla_a \hat{Q}_{UB}(s,a)\right]_{a=\mu_T}||_{\Sigma_T}} \quad (3)$$
$$\Sigma_E = \Sigma_T$$

3.3. Bang-Bang Control

We call a policy, which samples actions close to or at the action space boundaries a bang-bang policy (Bellman et al., 1956). This emphasizes the fact that a bang-bang policy technically only has two actions: the upper limit of the action space and the lower limit of the action space. This is undesirable, because, on one hand, deploying such a policy to a real robot system would result in very high torques which results in high wear and tear. Additionally, an operation with high torques would be dangerous to people standing close to the robot. On the other hand, a bang-bang policy is undesirable intellectually. A bang-bang policy is very simple to implement. Therefore, our sophisticated RL algorithms would be useless, if all we need was a bang-bang policy.

In OAC, the exploration policy is obtained by optimizing the objective within a given trust region as described in Equation 2. Because the underlying optimization objective has a linear form, the length of the mean shift of the policy is determined by the constraint and in particular by the maximum KL divergence δ as mentioned in Equation 3.

While OAC better explores the state action space and leads to a slightly better performance compared to SAC (Ciosek et al., 2019) as shown in Figure 3, with an arbitrarily large maximum KL divergence, the trust region can be expanded in such a way that the policy results in a bang-bang policy. By design this makes OAC very sensitive to good hyperparameter choice and requires careful hyperparameter-tuning. Additionally, as discussed, this makes OAC difficult to deploy to a real robot system. Although there are several methods to smooth exploration when applied on a real system (Raffin et al., 2022), having a bang-bang policy makes it even more difficult.

This can be examined by looking at the sampled actions that are used to explore the environment.

In addition to the one-legged jumping robot *Hopper*, the experiments were also carried out on the two-legged cheetah robot *HalfCheetah* and the four-legged walking robot *Ant* of the Multi-Joint dynamics with Contact (*MuJoCo*) physics engine (Todorov et al., 2012). All agents are trained for 1.5 million steps in the environment and further hyperparameters can be found in Appendix B.

All actions that are used for exploring the environment during training are used to estimate the probability density represented as a histogram in Figure 1 for the *Hopper* and in addition in Figure 6 and Figure 7 for the other two environments *Ant* and *HalfCheetah*. The action densities are plotted for each dimension individually to evaluate the action spaces independently and averaged over 5 random seeds.

Especially in the *Hopper* and *HalfCheetah*, the tendency of OAC to generate a bang-bang control can be clearly seen, since the actions at the action boundary have the highest density. Although SAC also has a tendency to use actions for exploration that are at the action limits, OAC has a much more dominant behavior. Even in the *Ant* the tendency of OAC to sample actions at the action limit can be clearly seen, but not as distinct as in the other two environments.

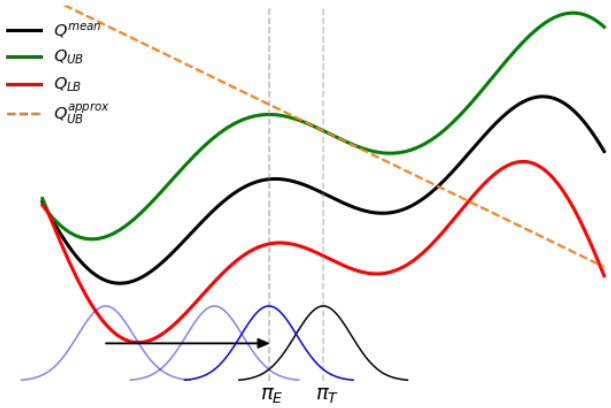

Figure 2. The upper confidence bound of the Q function is linearly approximated. Q^{mean} refers to the "true" Q function computed by the mean of both state-action value approximations, whereas Q_{LB} and Q_{UB} indicate the lower and upper confidence bound. μ_T is the target policy around which the upper bound is approximated. μ_E is the exploration policy.

4. Trust Region Optimization of Optimistic Actor Critic

4.1. Idea and Motivation

In general, OAC achieves a better performance than SAC using an extended exploration strategy by maximizing an upper confidence bound. But as stated above, sampling actions at the action limit results in a bang-bang like control and if OAC is applied in a real environment it could damage the agent or it makes it difficult to combine it with strategies for learning in real world environments.

This problem can be addressed by improving the sensitivity of the hyperparameter δ in Equation 2. One possible way could be an automated adjustment of the KL-divergence similar to the automated entropy adjustment in SAC (Haarnoja et al., 2019). Since this would not solve the problem directly, it can be obtained by, e.g., decreasing the trust region with increasing number of steps in the environment. Therefore, it could lead to smaller mean shifts for each action sampling process and accordingly to a reduced probability of sampling actions near the action limits. As this does not tackle the main issue, especially in the first few training steps, another option is used to adapt the optimization problem to be more robust to different hyperparameter values and guarantee at least the same exploration properties.

Trust Region Optimization methods are one of the most important numerical optimization methods and usually uses a quadratic model to approximate the original objective function in the region around the current best solution (Sun & Yuan, 2006). OAC corresponds to Trust Region Optimization without evaluating the quality of the intermediate solution and therefore uses only one iteration of improving.

In certain circumstances it can also be shown that line search methods are special variants of trust region methods and therefore improving the intermediate steps using backtracking line search while satisfying the trust region constraint is an efficient way to obtain a fast and good optimization solution. In terms of policy optimization, this has been successfully applied in *Trust Region Policy Optimization* (TRPO) (Schulman et al., 2015), where the Natural Policy Gradient (Kakade, 2001) is extended by backtracking line search. It can be shown that OAC, which is not used to update the target policy but to generate a local exploration policy, resembles the Natural Policy Gradient.

The extension of OAC by using multiple iterations to improve the objective, and in this case the upper confidence bound Q_{UB}, can be seen in Figure 2. In order to stay computational efficient and obtaining a closed form solution, the objective is linearly approximated, as done in OAC. To improve the intermediate results of the trust region optimization, the upper confidence bound is evaluated and afterwards backtracking line search is applied. Therefore, the first iteration, which corresponds to the exploration policy obtained by OAC and determined by the KL-divergence constraint, is the most left policy colored in slightly transparent blue. Evaluating the UCB, backtracking line search with a fixed step size reduction of a half is applied. This process is terminated when a higher UCB value at the exploration mean compared to the value at the target policy mean is achieved.

Whereas OAC guarantees to sample actions with a higher approximated UCB value, using backtracking line search generates actions with a higher real UCB value.

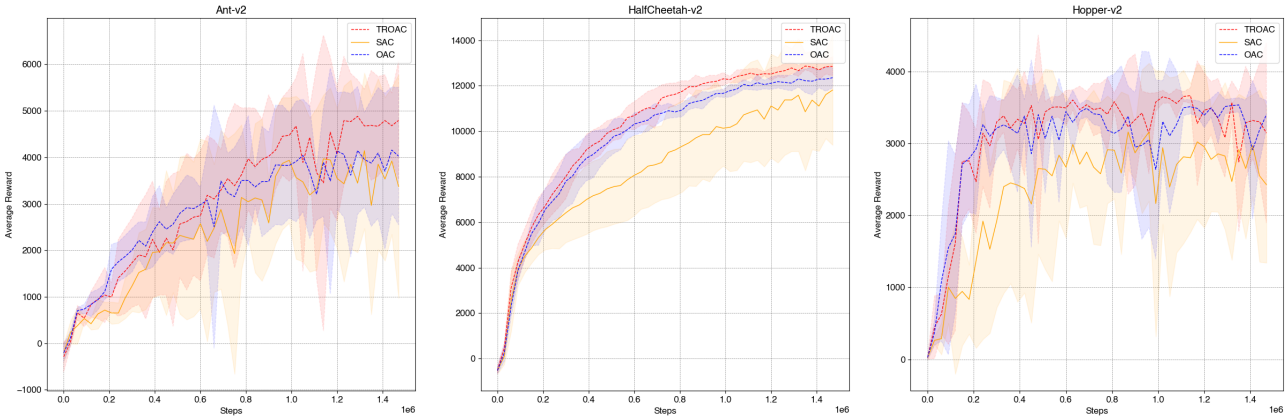

Figure 3. Comparision of SAC, OAC and TROAC, evaluated in *Ant*, *HalfCheetah* and *Hopper* environment for 5 random seeds and hyperparameters described in Appendix B. The x-axis shows the steps in the environment while the y-axis corresponds to the average reward R. The mean and the two-sigma confidence interval are visualized.

4.2. Derivation

Combining better exploration behavior of OAC and the benefits of backtracking line search applied on trust region optimization, we introduce the novel algorithm *Trust Region Optimistic Actor Critic* (TROAC). As described in 4.1, the UCB objective in Equation 2 is maximized resulting in better actions for exploration. To evaluate the actual improvement of the upper confidence bound Q_{UB}, we compare its value at the mean of the exploration policy μ_E with its value at the mean of the target policy μ_T. If this is the case, this results in the same exploration policy as OAC and the maximum update length is applied. Otherwise we perform a backwards line search by moving the mean of the exploration policy closer to the mean of the current policy and therefore reducing the length of the mean shift between the target and exploration policy. We iterate this procedure until we either obtain an exploration policy, which has a higher upper confidence bound value than the current target policy, or the exploration policy converges to the target policy. To ensure determinism, the algorithm is limited to K iterations. After that, the target policy is used for exploration. In each iteration the reduction of the length of the mean shift is controlled by the hyperparameter $\alpha \in (0, 1) \subset \mathbb{R}$. Keeping the variance of the current policy fixed, the exploration policy of TROAC is described as

$$\mu_E = \mu_T + \alpha^j \sqrt{2\delta} \frac{\Sigma_T \left[\nabla_a \hat{Q}_{UB}(s,a)\right]_{a=\mu_T}}{||\left[\nabla_a \hat{Q}_{UB}(s,a)\right]_{a=\mu_T}||_{\Sigma_T}} \quad (4)$$
$$\Sigma_E = \Sigma_T,$$

where $j = 0, 1, .., K$ refers to the j^{th} iteration of the line search. The pseudo-algorithm can be found in Algorithm 1.

By design, TROAC has an exploration policy that results in equal or better upper confidence bound values than the target policy μ_T and additionally, it tends to sample actions that are closer to the target policy than OAC.

5. Experiments

TROAC is evaluated in comparison to SAC and OAC for the one-legged jumping robot (*Hopper*), the two-legged cheetah robot (*HalfChetah*) and the four-legged walking robot (*Ant*) in the environments of the Multi-Joint dynamics with Contact (*MuJoCo*) physics engine (Todorov et al., 2012), similar to the experiments described in section 3.3. All agents are trained for 1.5 million steps in the environment with 5 random seeds and, for a better comparability, OAC and TROAC are implemented using the same hyperparameters that can be found in Appendix B.

5.1. Performance

The performance can be compared using the average reward in each episode shown in Figure 3. According to (Ciosek et al., 2019), OAC outperforms SAC in all three environments. Especially in the first iterations the benefits of OAC can be clearly seen, even if OAC does not guarantee a better exploration policy due to the linear objective of the trust region optimization, which can result in worse actions. This can be justified by the additional randomness of OAC, which is encoded in the optimistic upper bound and is also applied in TROAC, since unexplored and low-expected Q-function values can also lead to faster exploration of the state-action space and thus to an increase in performance.

Based on the results, TROAC seems to perform better than OAC due to the guarantee of sampling actions with a higher

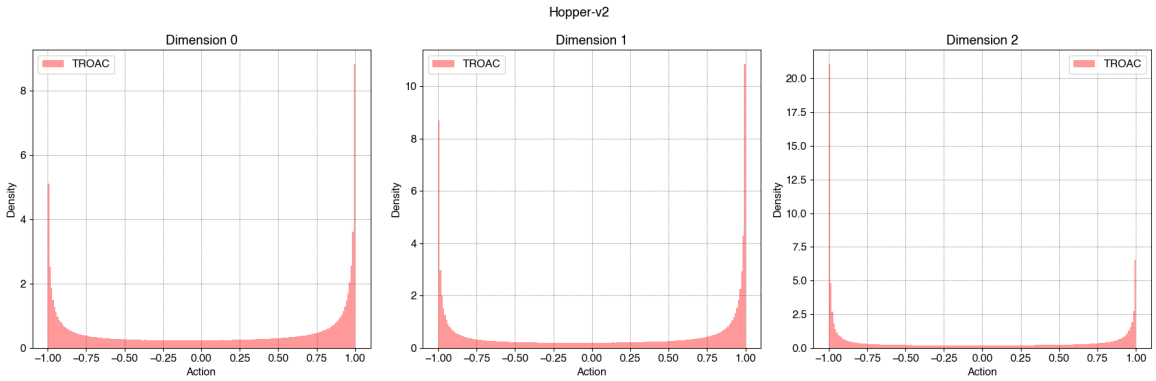

Figure 4. Histogram of sampled actions of TROAC, evaluated in *Hopper-v2* environment for 5 random seeds and hyperparameters described in Appendix B. The x-axis show the action space while the y-axis corresponds to the density.

upper confidence bound value. In the *HalfCheetah* environment an improvement can be clearly noticed. Although it seems to perform equally good in the *Ant* environment, it is hard to distinguish if TROAC leads to better results in the *Hopper* environment.

Overall, this shows that TROAC encodes at least as good exploration behavior as OAC and tends to outperform it. This confirms the proposition in Section 4.1.

5.2. Action density

In relation to the considerations about the problem of sampling actions near the action limits in Section 3.3, we evaluate the sampled actions generated by TROAC in Figure 4 compared to the corresponding diagrams for SAC and OAC in Figure 1. While only the *Hopper* environment is shown here, the action densities for the *Ant* and *HalfCheetah* can be found in Appendix A.

At first glance it does not look like TROAC will improve the sensitivity of the KL divergence constraint limit, but a closer look at the scaling reveals a difference in magnitude although not in shape. TROAC clearly reduces the density of actions near the boundary and increases the density mass between the action limits. This can be seen in the *Hopper* and *HalfCheetah* environment. Although a bang-bang like control exists for TROAC, it is not quite as pronounced as for OAC. However, in relation to the performance in Figure 3, that does not mean that TROAC behaves worse in exploration.

Especially the result of TROAC in the *Ant* environment show the trade-off between sampling actions at the trust region boundary, as done in OAC, and sampling pessimistic actions, as generated by SAC. For example in the first and fourth action dimension in Figure 6 for SAC, both densities have a clear maximum. In the case of OAC the density is much flatter and accordingly TROAC interpolates between these two, because the backtracking line search in Equation 4 enables TROAC interpolating between both extremes.

5.3. Discontinuity cost

In relation to the action density, a bang-bang like control can also be evaluated in terms of the discontinuity cost between two consecutive actions a_t and a_{t+1}:

$$c_t = ||a_{t+1} - a_t||_2 \tag{5}$$

In Figure 5 the discontinuity cost is averaged over each epoch for the *Ant*, *HalfCheetah* and *Hopper* environment.

While the lowest cost is obtained by the pessimistic exploration of SAC, OAC performs the worst and generates the highest change in consecutive actions for all three environments. Similar to the results in Section 5.2 and without minimizing the discontinuity cost directly, TROAC naturally reduces the change of consecutive sampled actions. Combined with the performance results in Figure 3, TROAC improves the trade-off between optimistic exploration, resulting in better performance, and small discontinuities in actions, resulting in less bang-bang-like policies.

6. Discussion

6.1. Pessimistic Upper Confidence Bound

Nevertheless it could be expected that TROAC better interpolates between SAC and OAC and therefore generates more actions that are between the action limits.

Because the upper confidence bound in Equation 1 is computed using the empirical uncertainty estimate of the learned state-action value function approximations, it highly depends on the clipped Q-learning of the two bootstraps. In SAC this is done using the min double Q-trick (Haarnoja et al., 2018). Therefore, the upper confidence bound is pes-

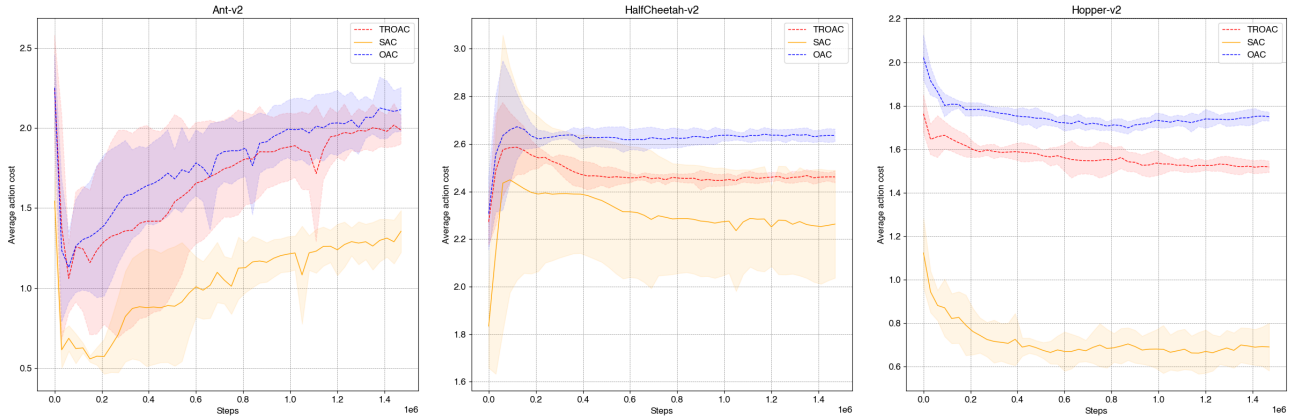

Figure 5. Comparision of SAC, OAC and TROAC, evaluated in *Ant*, *HalfCheetah* and *Hopper* environment for 5 random seeds and hyperparameters described in Appendix B. The x-axis shows the steps in the environment while the y-axis corresponds to the average action discontinuity cost in Equation 5. The mean and the two-sigma confidence interval are visualized.

simistic by design, since both Q function approximations are updated with the same target value. This is done by a shared minimum and accordingly uncertainty is only captured in state-action pairs, where no data is observed (Ghasemipour et al., 2021). Transferring these theoretical observation to optimistic exploration methods as OAC and TROAC that rely on a good upper confidence bound estimation, the observed behavior of sampling actions near the action limits can be explained as follows.

Uncertainty, or in this case expressed as optimism, is higher in areas where no actions are sampled. With ongoing training success more and more state-action pairs are sampled such that the uncertainty within a specific region gets lower and the uncertainty outside of the region, where no state-action pairs are sampled, increases. Since the exploration policy encodes a Gaussian policy, the sampled raw actions are passed through a $tanh$ non-linearity to ensure that the actions are within the action limits, which is a common implementation technique and is also applied in the experiments above. Therefore, passing raw actions that fall outside a certain large region through the non-linearity always result in actions close to the action limits. This corresponds to the sensitivity problem described in Section 3.3. In combination with the fact, that the upper confidence at unobserved and thus uncertain state-action pairs can be higher than the value at observed pairs, this leads to an optimization problem, that has its maximum at or nearby the action limits.

This lack of a correct uncertainty estimate caused by stabilized Q-learning and reduction of overestimation, results in an incorrect upper confidence bound that affects the objective of the optimization in Equation 2. Accordingly, this potentially increases exploring actions near the action limits.

6.2. Future work

Having a good estimate of the UCB is essential for UCB-based methods like TROAC and OAC. As discussed, this is currently not the case.

In order to improve the upper confidence bound estimation and the described lack of correctness of the uncertainty estimation in Section 6.1, independent state-action value function approximations can be learned similar to *Model Standard-deviation Gradients* (MSG) (Ghasemipour et al., 2021). Closely related to this, the quality in terms of expressiveness of the upper confidence bound can be optimized by using an ensemble of Q-function approximations. Ensemble methods are already used for upper confidence bound exploration (Lee et al., 2021) (Chen & Kumar, 2020), but differ in how the target value is computed when each Q-function approximation is updated in the ensemble. While updates with shared targets contradicts the theoretical derivations in (Ghasemipour et al., 2021), using an ensemble of independent double-Q bootstraps seems very promising to bridge the gap between stabilizing Q-learning and ensuring correct uncertainty estimation. Accordingly, initial experiments by ensembles with a shared target value justify these ideas and the need for independent Q-function updates.

Besides improving the upper confidence bound estimation, the initial idea of automatically adapting the trust region as described in Section 4.1 can be used to globally constraint the action space such that the upper confidence bound estimations outside of this global region is not taken into account when the exploration policy is optimized. Therefore, exploration policies that result in very similar actions near the action limit due to the $tanh$ non-linearity will be avoided.

7. Conclusion

In this work, we introduce TROAC, an off-policy RL algorithm whose exploration policy interpolates between the exploration behavior of SAC and OAC. We have shown that OAC results in a bang-bang-like policy for the evaluated environments. We continued to show that TROAC reduces this problem, while still performing similar or even better than OAC. We highlight the problem of an pessimistic upper confidence bound of the Q-function and propose ideas to improve this problem. But the implementation of these ideas will be reserved for future work.

References

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

Bellman, R., Glicksberg, I., and Gross, O. On the "bang-bang" control problem. *Quarterly of Applied Mathematics*, 14(1):11–18, 1956.

Brafman, R. I. and Tennenholtz, M. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct): 213–231, 2002.

Carvalho, J., Tateo, D., Muratore, F., and Peters, J. An empirical analysis of measure-valued derivatives for policy gradients. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10. IEEE, 2021.

Chapelle, O. and Li, L. An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, 24:2249–2257, 2011.

Chen, S. and Kumar, R. Efficient exploration via actor-critic ensemble. Dec 2020. URL https://www.sihao.dev/assets/CS287_Report.pdf.

Ciosek, K., Vuong, Q., Loftin, R., and Hofmann, K. Better exploration with optimistic actor-critic, 2019.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Ghasemipour, S. K. S., Gu, S. S., and Nachum, O. Why so pessimistic? estimating uncertainties for offline rl through ensembles, and why their independence matters. 2021.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft actor-critic algorithms and applications, 2019.

Hasselt, H. Double q-learning. *Advances in neural information processing systems*, 23:2613–2621, 2010.

Kakade, S. M. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017.

Lee, K., Laskin, M., Srinivas, A., and Abbeel, P. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 6131–6141. PMLR, 2021.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Raffin, A., Kober, J., and Stulp, F. Smooth exploration for robotic reinforcement learning. In *Conference on Robot Learning*, pp. 1634–1644. PMLR, 2022.

Russo, D. and Roy, B. V. Learning to optimize via information directed sampling. *CoRR*, abs/1403.5556, 2014. URL http://arxiv.org/abs/1403.5556.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

Sun, W. and Yuan, Y.-X. *Optimization theory and methods: nonlinear programming*, volume 1. Springer Science & Business Media, 2006.

Sutton, R. and Barto, A. Reinforcement learning: An intro, 1998.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Whitney, W. F., Bloesch, M., Springenberg, J. T., Abdolmaleki, A., Cho, K., and Riedmiller, M. Decoupled exploration and exploitation policies for sample-efficient reinforcement learning. *arXiv preprint arXiv:2101.09458*, 2021.

Supplementary Material

A. Action density evaluation

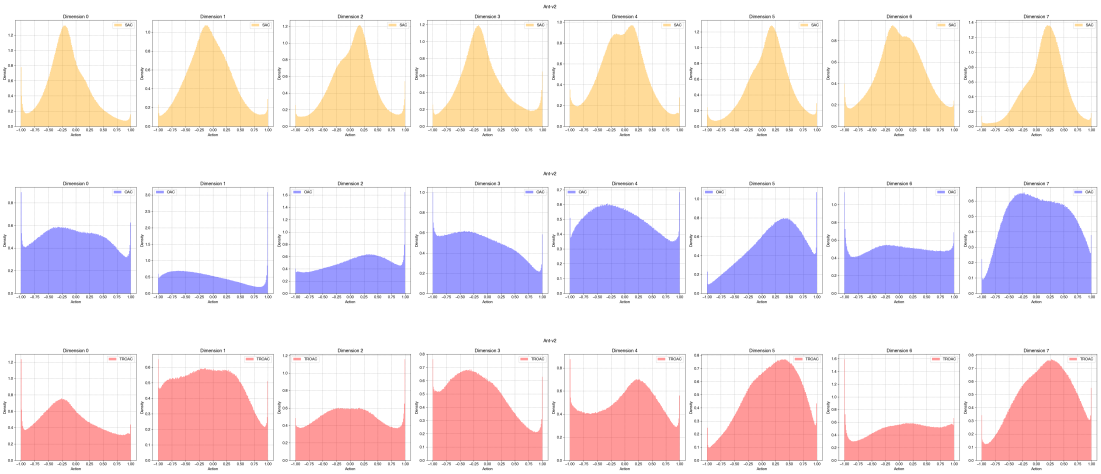

Figure 6. Histogram of sampled actions of SAC, OAC and TROAC, evaluated in *Ant-v2* environment for 5 random seeds and hyperparameters described in Appendix B. The x-axis show the action space while the y-axis corresponds to the density.

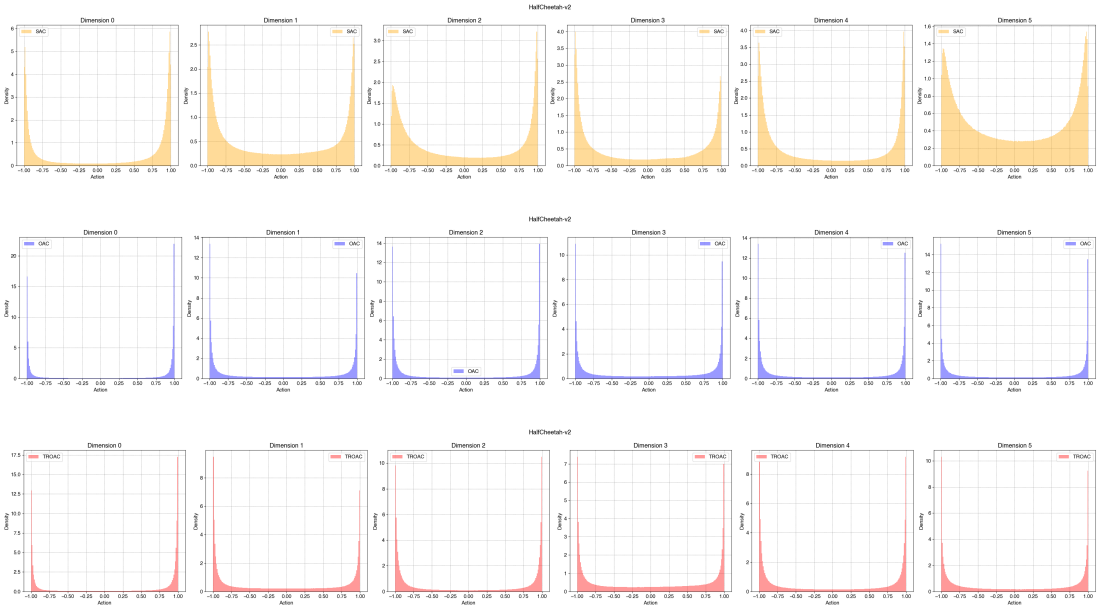

Figure 7. Histogram of sampled actions of SAC, OAC and TROAC, evaluated in *HalfCheetah-v2* environment for 5 random seeds and hyperparameters described in Appendix B. The x-axis show the action space while the y-axis corresponds to the density.

B. Hyperparameters

The following hyperparameters are used for the MuJoco environments *HalfCheetah-v2*, *Ant-v2* and *Hopper-v2*, that are used to evaluate the different algorithms in the result plots.

Table 1. SAC Hyperparameters

HYPERPARAMETER		VALUE
HORIZON		100
DISCOUNT FACTOR	γ	0.99
TARGET SMOOTHING COEFFICIENT	τ	0.005
EPOCHS		50
STEPS/EPISODE		3000
EPISODES EVALUATION		10
BATCH SIZE		256
WARMUP TRANSITION		10000
MAX REPLAY SIZE		10^6
CIRITIC NETWORK		[256, 256] RELU
ACTOR NETWORK		[256, 256] RELU
OPTIMIZER		ADAM [1]
LEARNING RATE ACTOR	α_π	3×10^{-4}
LEARNING RATE CRITIC	α_Q	3×10^{-4}

Table 2. OAC Hyperparameters - extending the hyperparameters of SAC in Table 1

HYPERPARAMETER		VALUE
SHIFT MULTIPLIER	δ	23.5298
UPPER CONFIDENCE BOUND	β_{UB}	4.66
LOWER CONFIDENCE BOUND	β_{LB}	-1.0

Table 3. TROAC Hyperparameters - extending the hyperparameters of SAC in Table 1

HYPERPARAMETER		VALUE
BACKTRACKING REDUCTION COEFFICIENT	α	0.5
MAXIMUM BACKTRACKING STEPS	K	10

[1](Kingma & Ba, 2017)

C. Pseudo-Algorithm

Algorithm 1 Trust Region Optimization of Optimistic Actor Critic (TROAC)

Require: KL-divergence limit δ, backtracking coefficient α, maximum number of backtracking steps K
input Initial parameters θ_1, θ_2 of critics, ϕ of target policy

1: **for** each environment step **do**
2: Compute exploration policy using backtracking line search by optimizing Equation 2

$$\mu_E = \mu_T + \alpha^j \sqrt{2\delta} \frac{\Sigma_T \left[\nabla_a \hat{Q}_{UB}(s,a)\right]_{a=\mu_T}}{||\left[\nabla_a \hat{Q}_{UB}(s,a)\right]_{a=\mu_T}||_{\Sigma_T}}$$
$$\Sigma_E = \Sigma_T,$$

3: Sample action and store transition to replay buffer

$$a_t \sim \pi_E(a_t|s_t)$$
$$s_t \sim p(s_{t+1}|s_t, a_t)$$
$$\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t R(s_t, a_t), s_{t+1})\}$$

4: **end for**
5: **for** each training step **do**
6: Update critic bootstraps with parameters θ_i by clipped Q-learning (Haarnoja et al., 2018)

$$\hat{\nabla}_{w_i} ||\hat{Q}^i_{LB}(s_t, a_t) - R(s_t, a_t) - \gamma \min(\check{Q}^1_{LB}(s_{t+1}, a), \check{Q}^2_{LB}(s_{t+1}, a))||^2_2$$

7: Update policy with parameter ϕ by $\nabla_\phi \hat{J}^\alpha_{\hat{Q}'_{LB}}$ (Haarnoja et al., 2018)
8: Update target network parameters by exponentially smoothing

$$\check{\theta}_1 \leftarrow \tau\theta_1 + (1-\tau)\check{\theta}_1$$
$$\check{\theta}_2 \leftarrow \tau\theta_2 + (1-\tau)\check{\theta}_2$$

9: **end for**
output θ_1, θ_2, ϕ {Optimized parameters}